Smart contracts in the operation and maintenance industry
a transaction cost economics and contracting theory perspective

Hamers, E.H.W.

Award date:
2019

Smart Contracts in the Operation and Maintenance Industry

A Transaction Cost Economics and Contracting Theory Perspective

Master thesis for MSc Innovation Sciences,
Eindhoven University of Technology
Faculty IE&IS

Author:
Lisenka Hamers
e.h.w.hamers@student.tue.nl
student number 0843105

Key words:
Blockchain; Ethereum; Smart contract; Transaction Cost Economics; contracting; contracting theory; operation and maintenance

1st Supervisor:
B. M. Sadowski, faculty IE&IS
b.m.sadowski@tue.nl

2nd supervisor:
F. Alkemade, faculty IE&IS
f.alkemade@tue.nl

3rd supervisor:
C. Castaldi, faculty IE&IS
c.castaldi@tue.nl

Company:
Arcadis Nederland B.V.

Company coach:
M. Adriaanse
martijn.adriaanse@arcadis.com

Abstract

Blockchain technology is a term well-known throughout business since its arrival with Bitcoin in 2008. However, even though it is considered to be a revolutionary technology that will change the way business is done, practical business applications are still missing. A new term has recently surfaced in combination with blockchain: smart contract. Smart contracts are a collection of conditional statements that can manage digital assets. From a business perspective, the question has risen whether the technology can be used for contracting. Two types of smart contracts can be distinguished: code-only and ancillary. Code-only smart contracts solely exist electronic, and encompass the entire contract. Ancillary smart contracts also have a text-based counterpart that covers non-automatable aspects of a contract. A text-based counterpart is useful for an increased flexibility to deal with necessary changes and circumstances during (especially long duration) projects. I expect that ancillary smart contracts will be easiest to adopt by firms, since it mostly provides automation, but code-only smart contracts are overall the better choice to grow towards, and should be the goal to implement.

The main research question that this thesis answers is: to what extent can blockchain-based smart contracts be used in Ethereum for contracting in the infrastructure division of the operation and maintenance industry? Interviews with key individuals were done in order to find how contracts in this industry division look like. Two extremes were found: technical contracts, and performance contracts. Transaction Cost Economics focuses on how to decrease opportunism in transactions. And while complete contracts are theoretically considered to be the best solution for opportunism in transactions, contracts used are generally incomplete. Then, using Contracting Theory, it was attempted to describe smart contracts and contracts in the infrastructure division of the operation and maintenance industry in identical terms. It was found that code-only smart contracts and technical contracts can be considered complete contracts, and are thus most likely to become possible to be written as smart contracts. While ancillary smart contracts and performance contracts can be considered incomplete contracts, with very little chance of being written as a complete contract. From this perspective, a business should aim to implement code-only smart contracts to minimalize opportunism, but the actual possibility of doing so is limited by transaction costs.

The main findings are that blockchain-based smart contracts can be used to solve communication and standardisation issues in the targeted industry division. Making use of blockchain-based smart contracts on Ethereum is not recommended for this industry, because transactions are solely business-to-business. The privacy that firms desire for contracting is more difficult to achieve in a public, permissionless blockchain network like Ethereum. Instead, it is recommended to make use of an alternative blockchain technology, like a private, permissioned blockchain, where the users are known and data is only available to those who have permission. But getting the government to also operate on the network will require an increase in transparency, making a public blockchain network more desirable. Overall, finding a balance between transparency and privacy will be key for the implementation of the technology. The governance structure of the network will be important as well, as centralisation is desired for speed, but not security. And decentralisation is desired for security, but then the speed of the network will decrease.

Foreword

When I was looking for a thesis subject, for me the most important thing was that I would have an application for my subject, an empirical aspect to the research. I did not want to just delve into books and approach a subject purely from a theoretical perspective. So, I went looking for a company. After some applications I ended up with Arcadis Nederland B.V. Here, they wanted to know more about blockchain technology, and whether they could do anything with it. Because I thought blockchain technology was an interesting technology, of which I had heard both a lot of positive and a lot of negative remarks, I was quite excited to work with it. Finding a subject that was academically relevant though, was still quite difficult.

My thesis supervisor, Bert Sadowski, helped me incredibly in identifying a proper subject, and setting me in the right direction. I know I can be a bit on my own: I tend to want to have a semi-finished product before I ask for feedback. This is simply because I feel like a lot of feedback tends to come down to: 'you should still explore this area' when you ask for it too soon. And I have had times where writing seemed impossible. Where it felt like I was just butting my head against a brick wall, not even making a dent. Other times I would suddenly write a whole lot of text. I would get extremely nervous before interviews, and I feel like I definitely messed up during one a few times. Overall, the process of writing this paper felt like a steady pace forward. Sometimes I would get a few bigger steps in, other times the pace was barely a crawl, but I was always trying to move forward. A thesis is not an end project to be easy, so I did not expect it to be. However, working on the same project for a few months definitely showed me how I was not used to doing such a thing. A master thesis feels like doing a group project, but on your own, and it has definitely been the most challenging thing I have done during my time as a student. But I suppose it is supposed to be just that.

I hope this thesis will help the reader understand blockchain technology, smart contracts, Transaction Cost Economics, contracting theory, etc. And that it helps the reader find some inspiration in these areas, should he/she desire so. And hey, if someone can explain blockchain technology past the 'well, you have blocks and they are chained together, and it saves transactions on these blocks', then I already feel like I did something right.

Lastly, I want to thank everyone for helping me write this thesis. I know this sounds sappy, but I really appreciate all the help and support that I received over these past five months, both on the professional and personal level.

Contents

1. Introduction

Blockchain is a new technology that made its entrance in 2008 with Bitcoin, a cryptocurrency. Blockchain technology is also called 'distributed ledger technology' or a 'trustless consensus machine' (Davidson, De Filippi, & Potts, 2018; Kwak, Kong, Cho, Phuong, & Gim, 2019). These names refer to the technology's ability to reliably record transactions and create agreement regarding the ledger's state across the entire system, without the need for a trusted third party as an intermediary. Blockchain technology thus provides a way of doing transactions very different from how it tends to be done currently (e.g. sending money via a bank), it has the potential to change how we currently exchange in the digital environment.

There are four words you can generally pick up when a blockchain network is described: public, private, permissionless, and permissioned (Jayachandran, 2017; Xu et al., 2016). These words actually come in a set of two. A network based on blockchain technology is either private or public, and it is either permissioned or permissionless. The public or private aspect is dependent on who can see the data on the network. On a public blockchain, even people outside the network can look into the data. For a private network, data is shielded. To see the data, one has to be on the network, and needs explicit permission to see the data in the form of a key that 'unlocks' it. Permissioned versus permissionless is dependent on how one can join the blockchain network. For a permissionless network, anyone can join (you only need enough memory space left on your computer to install it). For a permissioned network, you need a certain qualification (or qualifications) before you can join. The decision whether you qualify lies with people on the network. This can be everybody on the network, but it can also be that only one particular individual (or a selected few) is (are) responsible for these decisions (Jayachandran, 2017). Appendix C gives a more detailed description of how blockchain technology works.

There are three main issues with blockchain technology for industrial usage: privacy; scalability; and lack of governance (Li, Sforzin, Fedorov, & Karame, 2017; Olleros & Zhegu, 2016; Tapscott & Tapscott, 2016). Privacy can be a problem if all data in the system is publicly available to all users in (and outside of) the system, as is in the case of public blockchain networks (Li et al., 2017). This is at odds with the common data sharing practices in industrial settings, where data is only shared with the intended stakeholders. Scalability is an issue because of throughput (how many transactions can the system register per minute). The more users in the system, the less throughput the system can manage. In case of a permissionless blockchain, it can scale to many users, at the cost of low throughput. In case of a permission-based blockchain, there is a better throughput, at the cost of the maximum number of users of the system. And finally, companies do not work in a democratic way like blockchain technology, therefore, the lack of central governance in blockchain networks can be undesirable for a company. Increasingly is looked at how to deal with these issues in blockchain technology. An example is Hawk, which is a framework for building privacy-preserving smart contracts (Kosba, Miller, Shi, Wen, & Papamanthou, 2016).

There are also many benefits to using blockchain technology, but it is mainly adept in building trust in the operation of the network. There are two reasons for this: the removal of a few nodes (users) will not disrupt the system as a whole (it will continue to function like before), and, more importantly, there is no need for an external third party to verify any transactions, or for you to trust your transaction partner (Nofer, Gomber, Hinz, & Schiereck, 2017; Puthal, Malik, Mohanty, Kougianos, & Yang, 2018; Tapscott & Tapscott, 2016). Especially the removal of verification by a third party is what is most expected to drive future trends in blockchain application (Nofer et al., 2017), and is what appeals to companies.

A firm wants to be efficient. It needs to be efficient, because it needs to produce something (service, product, anything) more efficiently compared to a situation where the firm does not exist (section 3.1 elaborates further on this statement). And when trying to be more efficient than its competitors, a firm will try and adopt a new technology or process to improve on its existing one. Trying to get and/or

maintain a competitive advantage is one of the main reasons for firms to innovate (Kandampully & Duddy, 1999). Blockchain technology is interesting from this perspective, because it is a relative new technology. For companies it is interesting to try and be the first to find a new application for it that increases their efficiency. Especially in the area of information technology, blockchain technology is explored for this purpose. Because it is such a new technology, with still many (unexplored) possibilities, firms hope that whatever they can try and do with it, will vastly improve their efficiency and/or business model.

1.1 Blockchain technology as an institutional technology

Blockchain as a new technology can be interpreted in two different ways. It can be considered an institutional technology, or a general-purpose technology (Bresnahan & Trajtenberg, 1995; Davidson et al., 2018). General-purpose technologies are characterised by their technological dynamism, and their potential for application in a large variety of sectors. They are enabling technologies: instead of offering final, complete, solutions, they enable more opportunities. Their usefulness is generally expressed in production costs. Institutional technologies are social technologies "*that have become a standard expected thing to do, given the objectives and the setting*" (Nelson & Sampat, 2001, p. 40). Institutions can also be considered to be 'the rules of the game'. For example, economic institutions are institutions that shape economic performance. When a new technology arises, it can become an institution if it provides a new, useful way of doing something (new). It can overlap, or possibly replace, an older institution. I want to argue that blockchain technology is an economic institutional technology, because it provides a new way of doing transactions, which are inherently an economic activity, compared to established economic institutions, like banks. Also, I think that describing the impact of blockchain technology on business processes purely in terms of production cost efficiency, as is the case if you consider it a general-purpose technology, excludes possible applications of the technology.

Then, assuming that blockchain technology is an economic institutional technology, there is the matter of how to analyse it. Another way in expressing how a technology becomes an institution is by looking at what costs it decreases. Institutional technologies decrease transaction costs, these are the costs of coordinating economic activities, or in other words: the costs of running the economic system (Nelson & Sampat, 2001; Williamson, 1991). This fits well with how blockchain technology changes the way in which transactions are done. Then, Transaction Cost Economics is a fitting perspective to use for analysing blockchain technology. The main argument of Transaction Cost Economics is that transaction costs are sometimes better dealt with on the market or within a firm, depending on what type of contract better fits the needs of the transaction (e.g. a long-term contract versus a new contract for every transaction). Thus, using Transaction Cost Economics to analyse blockchain technology, the question will quickly move towards contracting, and how blockchain technology can be used for this. Can companies use blockchain technology for transactions (contracting)?

Contracts and blockchain is a debate that is still ongoing. For example, the Ethereum White paper (the original paper, in which the principles and ideas behind the platform of Ethereum are described) describes that the platform is a way to code 'smart contracts' that move digital assets according to rules specified in them (Buterin, 2013). However, these 'smart contracts' do not resemble contracts as they are understood in the business environment. Contracting is a difficult and often inefficient process, with costs stemming from several sources (section 3.2 describes the process and costs more elaborately). Smart contracts can be useful to make the contracting process more efficient by, among other things, decreasing the costs of writing and enforcing the contract (see section 4 for more on this topic).

To get a perspective on how smart contracts function compared to actual contracts, it is necessary to compare the two types. Because there are infinitely many types of contracts, this thesis compares two extreme forms of contracts in the operation and maintenance industry: technical contracts and performance contracts. The operation and maintenance industry was taken as a focus, because the department I was in at Arcadis had its main focus on this industry. Therefore, finding experts to talk to

regarding the functioning of this industry would be best accessible, as well as possibly observing contracting processes if I desired to do so. By comparing the two extreme types of contracts in the operation and maintenance industry to the conceptual and abstract definitions of (smart) contracts, an extra empirical layer was added to make the results of the research more tangible. It also allowed for advice on how smart contracts can be used next to (or instead of) established contract forms.

Because Ethereum was created with an emphasis on the possibility to code smart contracts on it in a simple manner, I also mainly focus on the possibility of using smart contracts coded on this platform for contracting. This leads to the main research question: To what extent can blockchain-based smart contracts be used in Ethereum for contracting in the infrastructure division of the operation and maintenance industry?

A first sub-question can then also be defined as: why are complete contracts a solution to opportunism? (see section 3) While the terms used in this sub-question may not yet have been addressed, the relevance of asking this question will become clear in section 3.

A second sub-question then is: to what extent can smart contracts using Ethereum be defined in contracting theory? (see section 4)

A third sub-question is: to what extent can technical contracts and performance contracts be defined in terms of contracting theory? (see section 5)

A fourth sub-question is: to what extent do performance and technical contracts have determinants (assumptions and operational aspects) in common with smart contracts? (see section 6)

A fifth sub-question is: what are the managerial and technical consequences of applying smart contracts using Ethereum to technical contracts and performance contracts in the infrastructure division of the operation and maintenance industry? (see section 6)

A sixth sub-question is: what problem(s) do blockchain-based smart contracts solve in the infrastructure division of the operation and maintenance industry? (see section 6) This last sub-question is especially important for the potential application of blockchain-based smart contracts, because no technology will be adopted if it does not solve a problem. Or in business terms: there is need for a business case.

1.2 Scientific relevance

Current research is more concerned with either how smart contracts work, either their legal implications, or on how to deal with a certain problem with smart contracts (e.g. how to prevent that everyone can see what goes in and out of it). ln contrast, this paper is an attempt to view smart contracts from an economic/business perspective, by deriving determinants of these smart contracts for an empirical analysis of technical contracts and performance contracts. This paper is an attempt at analysing how blockchain technology can be used for cooperating, replacing and/or coordinating with other existing institutional methods in the contracting domain.

2. Methodology

This study is undertaken as an exploratory case study (Yin, 1992). An exploratory case study is used to examine a phenomenon in real-life context, when the boundary between the phenomenon and the context is unclear. The phenomenon this research focuses on is (blockchain-based) smart contracts. While these smart contracts are not yet implemented in real-life, there are several programs for managing and writing contracts that cover certain aspects of what a smart contract would do (this is further discussed in sections 4 and 5). Defining what a smart contract application would do in real-life context, required defining contracts on three different levels. On the theoretical level, contracting theory was used to abstractly define contracts. Then on the conceptual level, smart contracts had to be defined. And finally, on the empirical level, contracts in the infrastructure division of the operation and maintenance industry had to be defined.

2.1 Abstract and conceptual level

Academic sources were used for exploring the abstract and conceptual level. The TU/e website was used for finding appropriate databases. Information was needed concerning Transaction Cost Economics, contracting theory, blockchain technology, and smart contracts. Transaction Cost Economics and contracting theory are mostly discussed in the categories of economics, business, and management, while blockchain technology and smart contracts were mostly discussed in the categories of engineering electrical electronic, computer science information systems, and computer science theory methods. Because access to the databases had to be secured via VPN, not all databases were as easily accessible as others. These databases were used for the literature review:

- Web of Science
- Google Scholar
- Scopus

Transaction Cost Economics and contracting theory are no new fields to economics, but they find their origin in the same paper(s) and are still lacking a specific, universal definition. Because of this, articles were initially selected based on number of citations, in order to find the basis for both theories. To find whether any important recent work was done in their direction a different criterion was used next, which selected based on publication date (newest to oldest). Literature on both subjects does not know specific years of popularity in terms of number of publications, and a few academics are cited repeatedly throughout the literature.

Search terms used were:

- transaction cost economics (topic)
- transaction cost economics (topic) AND contract* (topic)
- transaction cost economics (topic) AND Williamson (author)
- transaction cost economics (topic) AND Williamson (topic)

- “contracting theory” (topic)
- “contracting theory” (topic) AND “complete” (topic)
- “contracting theory” (topic) AND “incomplete” (topic)
- “contracting theory” (topic) AND Hart (author)
- “contracting theory” (topic) AND Scott (author)
- “contracting theory” (topic) AND Hart (topic)
- “contracting theory” (topic) AND Scott (topic)

Contracting theory and transaction cost economics were taken as key search terms to ensure that results were concerning the two theories. Contract* was an added search term to transaction cost economics to

find whether there was an interplay between the two theories. Williamson for author was chosen because he has written a significant amount of literature on transaction cost economics, following the initial paper on the nature of the firm by Coase. For contracting theory key academics were Hart and Scott, both were involved in the writing of many papers concerning the subject and were highly cited. By adding these authors to the search terms for both contracting theory and transaction cost economics (as both authors and in the topic), I also filtered significant amounts of literature that were not relevant to my research, such as literature on nanotechnology.

Contrary to the economic theories, both smart contracts and blockchain technology are relatively new terms, with the oldest academic literature found dating from 1982. The literature has known a surge in recent years, with the years of 2017 and 2018 showing significant more results than previous years. This made for a significant amount of recent research that has often not yet had the chance to be cited or accumulate citations. Therefore, to find keywords for academic literature it was proven most efficient to go through blogs on blog.ethereum.org and move from there.

Search terms used were:

- blockchain (topic)
- blockchain (topic) AND contract* (topic)
- blockchain (topic) AND transparency (topic)
- blockchain (topic) AND advantage* (topic)

- smart contract* (topic)
- smart contract* (topic) AND contracting (topic)
- smart contract* (topic) AND contract* (topic)
- smart contract* (topic) AND transparency (topic)
- smart contract* (topic) AND advantage* (topic)
- smart contract* (topic) NOT blockchain (topic)
- smart contract* (topic) AND blockchain (topic)

The key terms contract*, transparency, and advantage* were taken for both smart contracts and blockchain, because for both terms it was desired to know how they were combined with contracts, what advantages and disadvantages are, and what literature says about their transparency. Contracting was added as a term to smart contracts, to see what literature stated specifically concerning the use of smart contracts for contracting. For smart contracts, it was also noted to see what results were given when it was combined with and without blockchain.

Overall, the most useful way of finding relevant literature was though snowballing. This means that a highly cited, more recent article was taken (after 2000), and its references were observed to find more relevant articles. Another way was taking a highly cited, but older academic article, and then searching what articles cited it. This proved effective in finding both critical reactions to the article as well as more recent literature regarding the subject that I was interested in.

Web of Science and Scopus were mainly used for finding leading articles, while Google Scholar proved most effective for finding articles when applying the snowballing method. This is because Google Scholar is not dependent on a single database of an academic search engine to find results, and was more likely to find the desired article.

2.2 Empirical level

For the empirical level of analysis, academic sources were insufficient to find results. This level required knowledge and experience in the industry, which was not described in academic sources. Therefore, to describe contracts in this level, a method was needed that allowed access to experiential knowledge.

This left qualitative research methods as opposed to quantitative research methods. Semi-structured interviews were deemed best to gain access to the desired knowledge, because these have a set number of questions that are typically asked between the different interviewees, while maintaining the freedom to ask additional questions when desired or necessary (Young et al., 2018). Appendix A contains the interview guide used during interviews.

An unfortunate incident erased any recordings done during interviews when they were not yet transcribed or coded, and there was no time left to redo all the interviews. Therefore the empirical evidence/analysis became dependent on the notes made by hand during each interview. Another method used to find additional information regarding contracts in the infrastructure division of the operation and maintenance industry was via academic search engines such as Google Scholar and Web of Science. This gave no results for contracts in the specific industry or division that this paper focuses on, but did allow for general descriptions of the contracts.

Interviewees were selected based on their knowledge and availability. To illustrate: a supervisor at Arcadis would be asked who would be good to talk to concerning *(subject X)*. I would then be referred to a person who could know more, or would know who to talk to. This process would continue until a person was found who had the required knowledge. This person would then be contacted to plan an interview should he/she be interested and available. A downside to this method is that it restricted the interviewees to people who worked at Arcadis Nederland B.V. To be more precise, the criteria by which interviewees were selected were:

- The person should have enough experience working in the industry to be considered a senior (this typically meant work experience of more than two years)
- The person should be considered an expert on at least one focus area as identified in the interview guide
    - Alternatively, a person could also be interviewed when showing substantial interest in an area of the interview guide, even when not having specific experience regarding the subject area. This was mainly useful regarding the focus areas of blockchain and smart contracts, since there was no official division that concerned itself in this area
- The person had to be working at, or involved with, the department that operated in the infrastructure division of the operation and maintenance industry

Table 1 shows an overview of the interviews. They are placed in chronological order. Some people were interviewed multiple times, these are then represented multiple times in the table. Some interview notations were not officially planned interviews, but since relevant information was provided, they were still noted.

Table 1 – Overview of the interviews done in chronological order, giving ate, interviewee, company, and function

	Date	Interviewee	Company	Function
1	1-10-2018	C. Beltman	Arcadis Nederland B.V.	Cost expert
2	12-10-2018	M. Zanen	Arcadis Nederland B.V.	Senior Advisor Asset Management Rail
3	18-10-2018	D.-J. Van driel	Arcadis Nederland B.V.	Head Advisory Group
4	18-10-2018	M. J. Grevink	Arcadis Nederland B.V.	Team leader and senior project leader
5	23-10-2018	M. Hartsema	Arcadis Nederland B.V.	Director Operate & Maintain infrastructure

6	24-10-2018	F. van Ruth	Arcadis Nederland B.V.	Senior project manager infrastructure
7	26-10-2018	J. van der Gaag	Arcadis Nederland B.V.	Senior advisor
8	31-10-2018	R. Snijders	Arcadis Nederland B.V.	Director market group Energy & Infra Objects
9	2-11-2018	P. Dourlein	Arcadis Nederland B.V.	Senior advisor Asset Management Rail
10	5-11-2018	R. Grosscurt	Arcadis Nederland B.V.	Assistant project leader, Asset Management Infra Objects & Installation
11	7-11-2018	F. van Ruth	Arcadis Nederland B.V.	Senior project manager infrastructure
12	7-11-2018	R. Opgenoort	Arcadis Nederland B.V.	Senior advisor Information Management
13	26-11-2018	J. Roling	IBM	Executive IT Architect, Global Blockchain Lead, Center of Competency Environment, Energy & Utilities
14	26-11-2018	R. Vrees	IBM	Managing Consultant Blockchain
15	27-11-2018	R. Maresca	Arcadis Nederland B.V.	Chief Information Security Officer
16	28-11-2018	R. Menten	Arcadis Nederland B.V.	Client Development director
17	29-11-2018	D. Stolker	Arcadis Nederland B.V.	Advisor in the department of Business Development
18	4-12-2018	J. Kooistra	Arcadis Nederland B.V.	Senior project leader
19	7-12-2018	R. Opgenoort	Arcadis Nederland B.V.	Senior advisor Information Management

The two interviewees from outside of Arcadis, were knowledgeable in the area of blockchain technology, and could tell more about how Hyperledger Fabric worked. Hyperledger Fabric is developed by, among others, IBM under the Hyperledger initiative by the Linux foundation.

2.3 Validity and reliability

The research methods in this paper are qualitative. The terms of validity and reliability originally stem from quantitative research, and the application of these terms to a qualitative research can be debated (Smith, 2003). I define reliability in qualitative research as the extent to which the research, should it be redone by different researchers, will produce the same results. To ensure reliability, I have included how I have chosen interviewees, and my general literature search methods. However, because of the relative newness of blockchain technology and smart contracts, it is very possible that the literature aspects of any repeated research will produce very different results, as the understanding and exploration of blockchain technology and smart contracts progresses.

Validity in qualitative research can be described as the extent to which results from the research can be generalised (Smith, 2003). The focus of this research was quite specific, mainly aimed at looking one specific industry division. However, this is for the empirical part. For the abstract and conceptual level the research results should be generalisable. Then, repeating this research for different industries should

show little difference in the conceptual and abstract levels, with the main differences in the empirical level.

3. Blockchain as an Institutional Technology: A Literature Review

In this section I will elaborate on the abstract theories of Transaction Cost Economics and contracting theory, which will be used as the lens through which relevant aspects of smart contracts are investigated. The sub-question that is answered here is: why are complete contracts a solution to opportunism?

3.1 Transaction Cost Economics

Because blockchain technology, as an economic institutional technology, is observed to lessen transaction costs and possibly change the way contracts are done, the theory of Transaction Cost Economics seems an excellent fit to further analyse blockchain technology. In this section the perspective of Transaction Cost Economics will be explored in relation to economic institutional technologies. Findings include how economic institutional technologies decrease transaction costs.

3.1.1 Why do firms exist

To understand transaction costs, it is important to understand that these exist, and why. If they didn't exist, there would not be different ways of coordinating economic activity, since there would not be a certain benefit of one mode over another (Williamson, 1979). So there would not be different economic institutions operating, and competing, with each other simultaneously.

From a pure theoretical perspective, markets operate using the price mechanism (Coase, 1937). This mechanism assumes that production is regulated by price movements, coordinated through a series of exchange transactions that determine production. In simpler words: production and price are determined by supply and demand. On the other hand, firms make use of an entrepreneur-coordinator who directs production. Why is the allocation of resources in some cases done via the price mechanism, and in other cases via the entrepreneur-coordinator? Put another way: why are there firms?

The simplest reason would be that firms exist because there are costs involved with the price mechanism that are not present (or less) in the environment of the entrepreneur-coordinator (Coase, 1937). The most obvious cost would be that under the price mechanism, there is a constant need to discover what the prices on the market are. Even if specialists emerge that could sell this information, these costs would not be completely eliminated. A second cost is that of negotiating and concluding a contract for every single transaction on the market. These costs can be heavily reduced in certain markets, but not eliminated. Firms can also not completely eliminate the costs associated with contracts, but they can heavily reduce these (Coase, 1937). Within firms, there is only one contract necessary between the one responsible for production and the entrepreneur who runs the firm. This contract is not especially specific as to what has to be done, it specifies the limits to which the producer has to listen to the entrepreneur (in exchange for monetary compensation, which can be either fixed or fluctuating). Thus, firms will likely arise in situations where short-term contracts are not satisfactory, and long-term, less detailed, contracts are preferred.

The view on the existence of the firm developed by Coase (1937) seems to answer the question on when a firm will grow and stop growing (defining growth as an increasing number of transactions within the firm). His reasoning follows that as long as a firm can do transactions cheaper than the market, the firm will continue to expand. As soon as the price for transactions within the firm becomes equal to the cost of transactions via the price mechanism of the market, the firm will stop growing. Likewise, when the transactions on the market have become cheaper than in the firm, the firm will shrink by doing the transactions via the market (again). The transaction costs of the firm will increase, because as the firm expands, it becomes less efficient (Coase, 1937). The decrease of efficiency of a firm as it expands, also explains why there are multiple firms on the market. It can be derived that centralization such as in a firm is only efficient on a certain scale, hence why multiple centralized nodes will co-exist, together forming a partly decentralized network.

A view stressed by Tirole (1999), Hart (1989), and Hart and Moore (1990) is the theory of property rights. This theory is similar to Transaction Cost Economics, but differs in that it looks at nonhuman, physical assets instead of human assets. This theory helps in explaining firm boundaries using the assets owned by a firm, but is mainly applied for explaining incentives for employees and the owner-managers in the case of mergers (i.e. two companies joining each other), where it matters which company will own the result of the merger. For example, if Audi merges with one of its suppliers, and Audi gains ownership of that supplier, the supplier's management will lose a significant amount of bargaining power. This is because in an independent situation the supplier's management maintains ownership of the production facilities, it can decide when to increase or decrease supply to Audi, threatening Audi with (for example) less access to the production facilities. On the other hand, if Audi after the merger owns the production facilities of the supplier, the supplier's management can only threaten to make their own labour unavailable (with this I mean the human labour of the supplier's management team), which has significantly less bargaining power compared to the former situation. Ownership of the production assets can also impact the supplier's management's incentive to invest in cost-saving or quality-enhancing innovations. The supplier will feel more incentive to do these innovations if it has the ownership, rather than when Audi has the ownership, since in the latter case, the producer will not feel its returns on investment. The ownership of nonhuman, physical assets can thus also lead to control over human assets, even if these can never actually be owned in the same sense of the word (Hart & Moore, 1990).

3.1.2 Transaction costs

Transaction Cost Economics places the most important difficulty of analysis on comparisons of transaction costs (Williamson, 1991). In the economic system, transactions take place on a continuous basis. On the market a contract is created for each transaction, while in a firm it is the norm to have one long-term, more flexible contract. Transaction costs are characterised by two additional properties: bounded rationality, and opportunism (Davidson et al., 2018; Ménard & Shirley, 2005; Williamson, 1979). Bounded rationality means that a person can only act as rational as his knowledge and ability to compute said knowledge allows for (in the time available for making the decision). It is impossible to avoid bounded rationality when there are humans involved in the process, because it is part of human limitation. Opportunism refers to the tendency of people to try and take advantage of circumstances. Especially opportunism is considered to be a central concept in Transaction Cost Economics (Williamson, 1979).

According to Transaction Cost Economics, the key attributes of transactions are asset specificity, uncertainty (unforeseen probabilities), and frequency (of dealings) (Ménard & Shirley, 2005; Williamson, 2008). Asset specificity refers to the extent to which the investments made to support a particular transaction have a higher value to that transaction than they would have if they were redeployed for any other purpose. The ideal transaction in law and governance would have an absence of asset specificity (Ménard & Shirley, 2005). To explain why firms exist in these terms: some transaction costs due to dealing with uncertainty, asset specificity, and frequency of dealings, are more efficiently done within firms than the market. Figure 1 gives an overview of the sources of Transaction Cost economics and the key attributes of transactions.

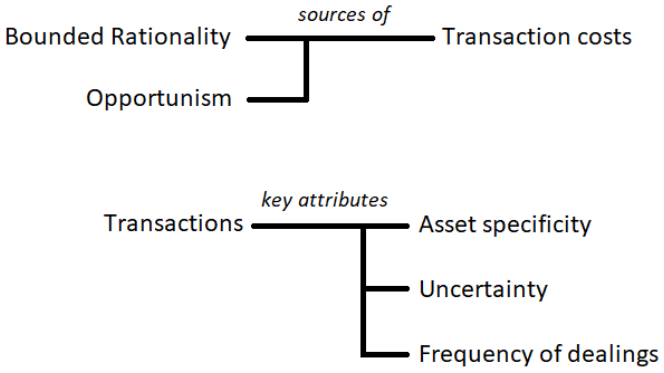


Figure 1 – Overview of the sources of Transaction Cost economics and the key attributes of transactions

3.1.3 Transaction Cost Economics on institutional technologies

Thus, how do Transaction Cost Economics and economic institutional technologies fit together? Institutional technologies (economic or not) impact transaction costs. To be more precise: an institutional technology is only adopted if it offers some form of transaction cost reduction compared to the currently used institutional technology. Therefore, it follows that from a Transaction Cost Economics perspective an institutional technology impacts transaction costs by decreasing opportunism. Therefore, in later sections it is important to address whether blockchain technology and smart contracts decrease opportunism, and if they do, how and why.

3.2 Contracting

Contracting is an important subject in Transaction Cost Economics, since it is implied that if there are no transaction costs contracting is costless (Williamson, 1979). This is partly supported by Scott and Triantis (2005), who state that contracts can protect (and encourage) specific investments, which are investments that are worth more when they support a particular transaction, than if they were redeployed for any other purpose (these are the investments measured in the concept of asset specificity). However, contracting is also a heavily debated subject. In this section contracting theory and the different types of contracts will be described and elaborated on.

3.2.1 Types of contracts

According to contracting theory, there are two different types of contracts: complete and incomplete (Hart & Moore, 1990; Scott & Triantis, 2005; Williamson, 1979). When a contract is complete, this means that it specifies obligations for each possible state of the world. Thus, there is no need for renegotiation between the parties. On the other hand, a contract is incomplete if it is not possible to account for all possible probabilities beforehand, which are consequently not recorded in the contract. Or, if the contract is open to multiple interpretations (Hart & Moore, 1990; Scott, 2003; Scott & Triantis, 2005). In both cases of incomplete contracts, renegotiations are then needed after the transaction. Transaction Cost Economics implies that bounded rationality causes all (complex) contracts to be incomplete contracts (Ménard & Shirley, 2005). Another reason for the incompleteness of contracts is that specifying all obligations in all states of the world can be too expensive (high transaction costs), which makes it infeasible to write a complete contract.

Even though definitions for the two types of contracts were given above to give an impression of what he two terms mean, there are no generally agreed to definitions regarding what a ‘complete’ and what an ‘incomplete’ contract is. In the next paragraphs the different definitions of incomplete contracting will be explored, and the generally agreed upon aspects. Then, complete contracting will be explored in the same manner.

Hart and Moore (1999) describe incompleteness as a case where the parties would like to add probability clauses to the contract, but are prevented from doing so by the inability to verify what kind of good should be traded, or because states are too expensive to describe after the transaction. State refers to two things: (1) the seller's costs of producing the traded good, and (2) the identification of the good to be traded and all other goods that are created, but not traded. They argue that the optimal contract is incomplete, and that the need to renegotiate makes a crucial difference compared to when no renegotiation is necessary. However, the describability of the to-be-traded good at the time of contracting is not important, provided that the good can be described after the transaction, meaning that the inability to describe goods after the transaction can make a large difference. Their conclusions rely heavily on the assumption that parties have to renegotiate (and to some extent that the parties will have to involve a third party). And on the assumption that in order to find the good that is to be traded, there are multiple other goods produced with slightly different functions, the one that is eventually traded is the one which gives the highest shared value.

Another way to describe incompleteness of contracts is by Scott and Triantis (2005), who say that a contract is incomplete when it fails to provide for the *efficient* set of obligations in each possible state of the world. Such a contract is informationally incomplete, but obligationally complete. This does make it a complete contract for lawyers, for whom it is important that all obligations are covered.

Finally, Tirole (1999) defines an incomplete contract as a contract that is vague, or silent on a number of key features, which causes it to not deliver the feasible outcome that the parties desire. In its implications this is closely related to the definition by Hart and Moore (1999), since both definitions emphasise that a contract is incomplete if it is found not sufficient after the transaction, and requires renegotiation or third party enforcement.

Then, what are the common aspects of these definitions of incomplete contracts? A general threat is that a contract is incomplete if, after the transaction, there are probabilities not specified in the contract that are necessary for enforcement of the contract without renegotiation. The cause of these missing contract clauses is across all definitions similar, and often comes down to a lack of information at the time of contracting, or the inability to write all information down. This lack of information can be because information is costly (and thus the information was not verified and added), or because the information is simply not yet available to the contracting parties. In all cases there will be renegotiations, to discuss how the contract should be changed considering the unspecified state. The transaction can even be cancelled if that appears to be the best for both parties. Scott (2003) makes the observation that contracts can also be incomplete by design: the contracting parties can choose to leave out certain obligations. Contracts that are incomplete because of this, tend to be unenforceable in court, because the court is not able to determine what the intended meaning of the contract clauses was, and is thus unable to provide an appropriate solution (Schwartz & Scott, 2003; Scott, 2003). To conclude, I define incomplete contracts as *contracts that have clauses that are vague, missing, or open for multiple interpretations, and that require renegotiation for enforcement after the transaction*. This definition encompasses the main aspects associated with, and the different definitions of, incomplete contracts discussed until now.

Now the question remains: what are complete contracts? Overall, there is a significant larger amount of literature regarding incomplete contracting than complete contracting. This is probably because discussing one also discusses the other, since complete contracts can somewhat be considered the opposite of incomplete contracts. Following the above definitions of incomplete contracts, complete contracts are contracts that are complete in the information they provide: it specifies obligations in each possible state of the world, with no place for ambiguity or multiple interpretations. Renegotiations are not necessary, because there are no states in which the contract doesn't specify the parties' obligations. To add to this by Scott and Triantis (2005), their definition of incomplete contracts suggests that a contract would be complete if there is no need for enforcement after the transaction. It also demonstrates

the difference between what a lawyer and an economist would call a complete contract. Personally I agree with Scott and Triantis (2005), and will define complete contracts as *contracts that are obligationally complete (and not per se informationally) of which there can be no discussion as to what the obligations mean, and which do not require renegotiation for enforcement*.

The next section will explain contracting theory and explain how complete and incomplete contracts are used. The assumptions behind the contracts will also be discussed, along with any problems and possible resolutions.

3.2.2 Contracting theory

Contracting theory is mainly aimed at analysing the transactions between firms, meaning that transactions between individuals, or firms and individuals, are outside of the theory's scope (Schwartz & Scott, 2003). Contrary to the popular belief that companies aim to maximise personal value, contracting theory assumes that the private goal of contracting parties is to maximise the shared value created by a contract (also called the surplus) (Schwartz & Scott, 2003; Scott & Triantis, 2005). This is because the potential benefit of a long relationship with multiple contracts can be larger than the gain a firm would get from exploiting the other firm, or breaching the contract. However, in the case of a single transaction, a company would aim more to increase its personal profit. Maximising the surplus in that case is only a goal insofar that it increases the company's profit.

The value created by an exchange is simple: when selling an object, it will have more value to the buyer than the seller. Because if the object is worth less to the buyer than the seller, then there will be no exchange. Then, however, there is the question of how engaging in a contract will create additional value to the parties. According to Scott and Triantis (2005), there are a few reasons for why parties might contract for future performance, but the most interesting reason (for contracting theory) is that "*one or both of the parties may be in a position to make investments in anticipation of the exchange that will increase the exchange value by either (a) lowering the cost of performance or (b) raising the benefit from performance*"(Scott & Triantis, 2005, p. 188). This reason is so interesting because it raises questions on how to balance the private and shared incentives of the contracting parties.

Economists call the optimal level of specific investments 'ex ante efficiency.' Then, 'ex post efficiency' describes the surplus of the exchange (Schwartz & Scott, 2003; Scott & Triantis, 2005). Ex ante efficiency can upset the ex post efficiency by compelling exchange when there is no surplus. Uncertainty regarding the cost and value of the contract at the time of contracting is considered the source of this. The parties thus seek to prevent that they get stuck in an unfavourable exchange (a contract can force them to commit to such an exchange). In the case of a wasteful exchange, this would be when the costs for the seller exceed the value of the product to the buyer. Then, the parties want to be able to renegotiate when it becomes clear that the surplus will not be enough to validate the transaction (Scott & Triantis, 2005). However, often one party has made costs that it cannot recover (these are called sunk costs), this makes the investing party vulnerable to exploitation of the other party. This can lead to the investing party reducing or even declining welfare enhancing investments, which in turn undermines ex ante efficiency. This phenomenon is also referred to as the hold-up problem (Schmitz, 2001). Ensuring ex ante and ex post efficiency is considered a difficult and important contracting problem in contracting theory (Schwartz & Scott, 2003; Scott & Triantis, 2005).

Economists who follow the property rights approach divide information costs (the costs responsible for the incompleteness of contracts) in two categories, based on two distinct stages of contracting (Hart & Moore, 1990; Schwartz & Scott, 2003; Scott & Triantis, 2005). The first category is the front end costs, also called (ex ante) transaction costs. These costs are illustrated in Figure 2. The costs stem from the fact that information is costly and sometimes unavailable to the parties *at the time of contracting*. The second category consists of the back end costs, also called (ex post) enforcement costs, which stem from the fact that information is costly and sometimes unavailable to the parties or the enforcing court *at the time of enforcement*. Another way to divide the costs in two stages is very similar. This reasoning

follows that in the back-end, there are renegotiation costs where each party tries to maximise his part of the total surplus (Hart, 1989). Then, because a party's share in the renegotiations does not have to be in relation to his investments, parties will have a wrong investment incentive during the front end (undermining ex ante efficiency). Both types of costs come down to the same principles: at the front end, there can be investment to enhance the surplus. However, the incentive to invest is dependent on the expected returns at the back end, which can be unknown at the time of contracting, or can appear to be inaccurate.

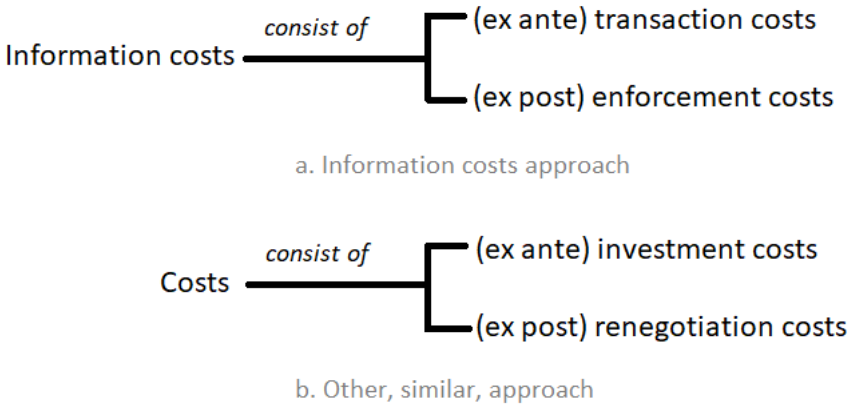


Figure 2 – The types of costs from 2 approaches

There are a few ways in which contracts can be enforced. There is enforcement by state, self-enforcement, and enforcement through reputational consequences (Schwartz & Scott, 2003). Enforcement by state refers to laws that help allocating risk between the two parties. State enforcement is especially important when investment is relation-specific, and when the realisation of a bad state of the world would give rise to significant disruption costs. With self-enforcing contracts, the threat of either party to no longer deal with the other is sufficient to induce performance. Reputational sanctions can work if a single party's boycott is not sufficient. However, in large economies particular contracting parties are often anonymous. This means that it is not easy for other parties to discover why the deal broke down. Therefore, reputations work best in small trading communities where everything that happens quickly becomes common sense throughout it, and boycotts are easily enforced.

3.2.3 Assumptions complete and incomplete contracts

Because of discussions regarding viability of assumptions, I want to give a short overview of the assumptions I consider to be behind complete and incomplete contracts. After the overview, I will first explain any assumptions found on incomplete contracts, followed by those of complete contracts. These parts can contain more assumptions than I will actually assume and state in the overview, because I can disagree with an assumption, or there can be too much conflict regarding it for me to incorporate it. For incomplete contracts, I assume the following:

(1) There is information symmetry,
(2) Parties behave rational, but the outcomes may not be describable,
(3) The state after the transaction can be observed, and always to some extent verified,
(4) Companies try to maximise their own profit, while maximising surplus, and
(5) There is always renegotiation.

For complete contracts I assume the following propositions:

(1) There is information symmetry,
(2) Parties behave rational, but the outcomes may not be describable,

(3) State is always observable, and always verifiable,
(4) Companies try to maximise their own profit, while maximising surplus, and
(5) There is no need for renegotiation.

In the next paragraphs I will explain the assumptions and any discussion regarding them.

3.2.3.1 Incomplete contracting

Incomplete contracting theory makes some more general assumptions. The first assumption is that the theory tends to assume information symmetry between the contracting parties (Hart & Moore, 1990; Schmitz, 2001; Tirole, 1999). The second assumption is that the contracting parties behave rationally (Hart & Moore, 1990), meaning that the probabilities are foreseen at the time of contracting, but not necessarily describable. This leads to the third assumption that the state after the transaction can be observed, and to some extent verified (Hart & Moore, 1990; Lind & Nyström, 2011; Scott & Triantis, 2005). That the state is to some extent verifiable means that it should be to some extent be observable for third parties, like a court. This assumption depends on the mechanisms that the parties have to renegotiate, and that the contractual possibilities are restrained by the renegotiation. The fourth assumption is that companies aim to maximise their individual profit, while also maximising the total surplus (Hart & Moore, 1990; Schwartz & Scott, 2003). The fifth and final assumption is that there is always a need for renegotiation (Ménard & Shirley, 2005).

The first assumption can be considered to be unrealistic: it is unlikely that both parties have the exact same information. The Principle-Agent theory can be used to observe incentives in the case of asymmetric information (Laffont & Martimort, 2001; Tirole, 1999). The theory finds that there are three types of information problems: the seller can do something behind the buyer's back (moral hazard), the seller can have information ignored by the buyer (adverse selection), and finally, after the transaction the state cannot be verified. The Principle-Agent theory is closely related to both contracting theory and Transaction Cost Economics, and shares its roots, which makes it an excellent addition to use in the case of information asymmetry.

The second assumption implies that all the payoff outcomes are also known at the time of contracting, since multiple possible contracts can give rise to the same outcome. This leads to the issue of describability: at the time of contracting, it might not be possible to describe the good that is to be traded (Tirole, 1999). The describability of the traded good was proven by Tirole (1999) to not matter in the case of incomplete contracts when it came to the possible outcomes. However, describability does matter when it comes to incentives for the investor, because the investor will not put in effort if he does not have the insurance that he will see his efforts in his payoff (will his effort be describable afterwards?).

The third assumption is heavily debated by scholars (Hart & Moore, 1990; Lind & Nyström, 2011; Scott & Triantis, 2005). Often, the assumption is actually made that the state can be observable, but not verifiable. However, I feel this is a strange assumption, and agree with Lind and Nyström (2011) that the state is always at least to some extent verifiable. And that it is very possible to build in mechanisms that make the state more verifiable in case it is suspected to be needed. Even though I admit that the state might not be verifiable to a level in which the third party can take a side, it should be able to observe some aspects of the state, even if it is not enough.

The fourth assumption does not always have to be fully accurate (Schwartz & Scott, 2003; Scott & Triantis, 2005). This assumption is based on the mechanism that companies always try and maximise their own profit. However, maximising the surplus is an important goal when companies attempt to establish a long-term relationship. Therefore, the assumption implies that companies always aim at making beneficial long-term relationships with the contracting partner, regardless of whether that will actually happen. This can also be explained by using reputation mechanisms: if a company always maximises its own profit at the cost of its contracting partner, at some point they will not have any

contracting partners left anymore, since no one wants to deal with them. Therefore, I agree with this assumption behind incomplete contracts. Companies will (ideally) aim at maximising their own profit, while maximising surplus, in order to maintain friendly relationships with contracting partners, and attract more potential contracting partners.

The fifth assumption follows form the incompleteness of the contract itself. The contract misses clauses or contains clauses with multiple possible interpretations. In order to deal with these, it is necessary for the parties to renegotiate whenever anything occurs that requires that. For example, an unexpected problem occurs, and the parties need to renegotiate in order to agree on a solution.

3.2.3.2 Complete contracting

Just like incomplete contracting theory, complete contracting theory also makes a few assumptions (Hart & Moore, 1990; Tirole, 1999). The first assumption is that all probabilities are foreseen and can be described (parties behave rational). The second assumption is that there is no need for renegotiation. The third assumption is that information is symmetric between the parties. The fourth assumption is that courts will always enforce contracts.

The first assumption has two possible implications: either describing probabilities is costless (ignoring front end information costs), or there are no wealth constraints. Both of these implications are unrealistic. However, not all definitions of a complete contract would necessarily have an impact using this assumption. For example, when a simple contract omits information and obligations that were found to not matter for the execution of the contract, the contract can still be considered complete: even if not informationally complete, it is still obligationally complete.

The second assumption is actually quite debated: not all complete contract theorists agree on this. For example, you can make a clause in the initial contract that states that renegotiation will be done, in order to find the best payoff distribution. In this case, if you have a renegotiation and change something compared to the initial contract, the initial contract was still complete, because it assumed this was necessary to happen. It is also argued that renegotiation is almost always beneficial to the parties (Tirole, 1999), and a contract that states that there will not be renegotiation could thus not be the best contract, even if it is complete in any other aspect.

The third assumption is debated as well. Often information symmetry is assumed because this is also assumed for incomplete contracting theory. And it was also proven that complete contracts are especially powerful in the case of symmetric information (Tirole, 1999). However, it can also be argued that in the case of symmetric information, collusion is more likely to happen. Collusion is a secret agreement between two or more parties to hinder competition, often through illegal means. This hinders the effectiveness of the complete contract in the environment where it should be strongest.

The fourth assumption should be more understood in terms of: courts should always be able to enforce a contract and are generally used to do so in case of conflict. This assumption boils down to the opposite of the assumption of ‘observable, but not verifiable’ that was discussed in incomplete contracting, since that inherently means that the contract contains clauses that cannot be enforced by a third party (because these cannot observe it). Perhaps it is better to summarise the first and fourth assumption into a single one: the state is always observable by a third party, and thus always verifiable.

I will assume a fifth and final assumption, which states that companies aim to maximise their individual profit, while also maximising the total surplus. I copy this from the incomplete contracting assumptions. While it is not specifically stated in the literature to be the case, it describes the motivation of a company when engaging in a contract, and there is no reason as to why this would be different for complete and incomplete contracts.

3.2.3.3 Comparison

Table 2 – Overview assumptions complete and incomplete contracts, the differences are made bold

Complete contracts	**Incomplete contracts**
Information symmetry	Information symmetry
The parties behave rational, but the outcomes may not be describable	The parties behave rational, but the outcomes may not be describable
State is **always** observable, and **always** verifiable	The state after the transaction **can be** observed, and **always to some extent** verified
Companies try to maximise their own profit, while maximising surplus	Companies try to maximise their own profit, while maximising surplus
There is **no need** for renegotiation	There is **always** renegotiation

Overall, the assumptions between the two theories are very similar. Table 2 shows the assumptions behind the two contracts side-by-side. The advantages of incomplete contracts lie in their guaranteed, complete observability, and the removed need for renegotiation. However, there are significant obstacles in the actual writing of a complete contract. Instead, the advantage that incomplete contracts hold is that they better represent the actual situation of contracts in use. Therefore, incomplete contracting theory is better suited for analysis.

3.2.4 Complete contracts as a solution to opportunism

Thus, incomplete contracts are a consequence of the conflict between ex ante and ex post efficiency problems, as defined by the property rights approach. The complete contract is often proposed as a solution to this conflict (Scott & Triantis, 2005). Either by compelling exchange in states where there is a surplus, or by directly specifying an obligation to make a certain investment, the complete contract encourages the correct amount of ex ante efficiency. Complete contracts will therefore also show from the very beginning whether a trade is profitable or not (because the outcomes are known). For example, when a complete contract shows that a trade has no benefits for both parties, the trade will never get past making the contract. However, front end information costs hinder writing down all the information and obligations that are necessary to consider a contract complete, even when you define a contract as complete when it is (only) obligationally complete (which saves costs because there is less need for information completeness). In other words, transaction costs hinder the actual writing of complete contracts in the business environment.

An alternative way to prevent exploitation of the investor is by allocating property rights to specific investments (Hart, 1989; Hart & Moore, 1990; Scott & Triantis, 2005). Then, bargaining power can be allocated according to property rights, this is done by looking at asset ownership (Hart, 1989). Ownership of an asset "*goes together with the possession of residual rights of control over that asset; the owner has the right to use the asset in any way not inconsistent with a prior contract, custom, or any law*" (Hart, 1989, p. 1765). It is important to note here that one cannot own human assets because these assets cannot be bought or sold, the human workers own them and will always maintain ownership of their own human assets. Then, what remains are nonhuman assets of which the ownership matters. The owner of the nonhuman assets can influence these on any area not specified in the contract, this gives him more bargaining power, and thus more influence on the division of the surplus. This would help ensure that the surplus will be more fairly distributed in renegotiations after the transaction, because bargaining power is distributed in a way to prevent the exploitation of one party. This option is attractive, since it does not require extra front end information costs and assumes that the contract is incomplete, and renegotiation is required.

Whether you observe purely from either the perspective of information costs or in terms of the property rights approach, making a complete contract will often be much more expensive than an incomplete contract, even if the complete contract is only obligationally complete. You can only judge whether a

contract was obligationally complete after the transaction, when it has become apparent whether the contract was satisfying, or whether renegotiation is required. However, when it comes to back end enforcement and renegotiation costs, the complete contract is significantly more efficient than incomplete contracts, or the property rights approach. In terms of dealing with opportunism, a complete contract does not allow it, while incomplete contracts and the property rights approach merely aim at minimising it, having accepted that it will always be a factor. Therefore, from a theoretical perspective, complete contracts are the most ideal way of dealing with opportunism.

Also, because I define complete contracts as contracts that are obligationally complete, in practice they will be less complete than it might suggest. For example, if the parties anticipate that something unexpected might happen of which they cannot yet tell the full impact, they can include in the contract that there will be a renegotiation as soon as something that was unexpected happens. This example ignores the assumptions in complete contracting that both parties act rational, and that there is no need for renegotiation. However, I think that when applying theory to reality, the realisation of all assumptions, be it in the case of complete or incomplete contracts, will often not apply and some flexibility is required.

Then, to take all we learned in this section about contracting, what should be analysed about blockchain technology and smart contracts? Blockchain technology itself cannot be analysed using contracting theory, because the scope of contracting theory is so specific. However, the theory is very useful for analysing smart contracts. A first important aspect to find about smart contracts, is whether they can be considered complete or incomplete contracts. This can be done by comparing the assumptions behind complete and incomplete contracts to the assumptions behind smart contracts. Contracting theory suggests that the best way to decrease/remove opportunism is by using complete contracts. Therefore, if smart contracts can be considered complete contracts, then they will (theoretically) remove/decrease opportunism.

3.3 Summary Transaction Cost economics and Contracting theory

In this section the main observations on transaction cost economics and contracting theory are summarised, along with observations obtained from joining the two theories.

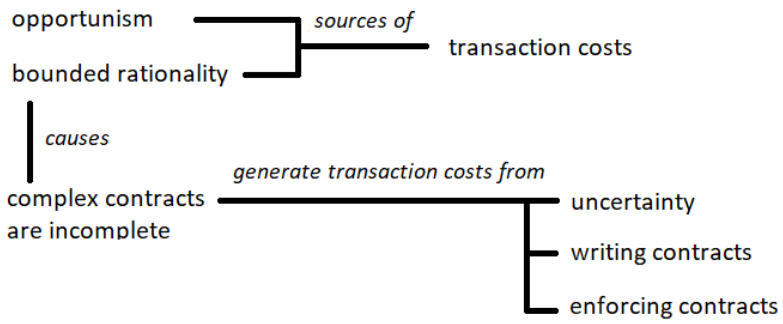
Incomplete contracts generally generate transaction costs out of three possible sources (Davidson et al., 2018; Hart, 1989; Tirole, 1999): uncertainty (unforeseen probabilities), costs of writing contracts, and finally, costs of enforcing contracts. Uncertainty is a consequence of bounded rationality: some possible future states could have been overlooked when writing the contract. Writing the contract costs time, and possibly information has to be bought. The more complete you want the contract to be, the more time and information it will take, and thus the more money is needed. Finally, the contract has to be honoured, enforcement can be done via a third party (like a court), or the contract can be self-enforcing (Schwartz & Scott, 2003; Tirole, 1999). An example of enforcement by a third party is that information is missing from the contract, and the court has to decide on the best solution to deal with the consequent problems. A more unfortunate version of this is that information is missing in a way that the court also cannot properly decide on the best solution.

These transaction cost sources are similar to the front- and back-end information costs associated with incomplete contracts. There are two similar views in this regard. The first view assumes that at the front end there are ‘ex ante transaction costs’, and at the back end ‘ex post enforcement costs’. The second view assumes at the front end investment costs, and at the back end enforcement costs. The mechanics between the costs are similar: depending on the expected relationship between the effort/investments made in the front end and the payout at the back end, a party has more or less incentives to invest in the front end. In other words, depending on the expected profit made from the ‘ex post efficiency’ in relation to the ‘ex ante efficiency’, there is more, or less, incentive to invest for the investing party. From all perspectives, especially unforeseen probabilities are considered a significant reason for the existence of incomplete contracts. Unforeseen probabilities can also be seen as the cause of negotiation and

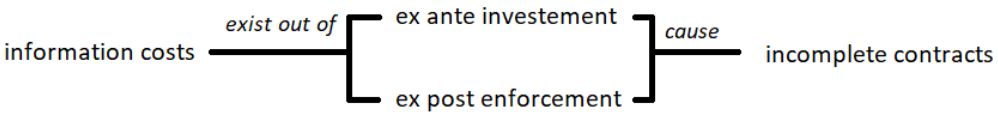
enforcement costs in the back end, since it can be argued that there is only a need for renegotiation or enforcement when it appears that something was missing in the initial agreement.

Overall, transaction cost economics and contracting theory have similar views, which is not surprising since both theories stem from similar foundations, like Coase's article 'Nature of the firm' from 1937, and Grossman and Hart's "The costs and benefits of ownership: a theory of vertical and lateral integration" from 1986. One aspect that differs between the two theories are the costs that are found to be the cause for incomplete contracts. Figure 3 gives a general overview of each theory.

Transaction Cost Economics argues that opportunism and bounded rationality have to be dealt with in other to minimise transaction costs. However, bounded rationality will always be present as long as there are humans in the process, therefore, in practice only opportunism can be minimised. Firms use incomplete contracts to deal with opportunism (Davidson et al., 2018; Hart & Moore, 1990; Williamson, 2008), Hart (1989) even describes companies as a collection of incomplete contracts (or 'a nexus of contracts'). The act of internalising transactions (and removing them from the market) is called vertical integration (Williamson, 1979). Thus, companies decrease transaction costs via vertical integration, essentially changing the ownership of any of the physical and human assets, but also reducing the investment risk for the investing party. This latter effect is because when a transaction occurs within the firm (it is integrated), both parties share the risks, because they have become (essentially) one party. In summary, incomplete contracts serve to internalise (and minimise) transaction costs, and deal with the hold-up problem (Hart & Moore, 1990).



a. Transaction Cost Economics



b. Contracting Theory

Figure 3 – Overview Transaction Cost Economics (a) and contracting theory (b) costs and causes

Both contracting theory and Transaction Cost Economics theory have weaknesses. Transaction Cost Economics can be considered a very negative view of the world, since it assumes that there is opportunism involved in every transaction. From this follows you can never fully trust your transaction partner. Depending on the industry, Transaction Cost Economics might not be the best choice for a framework, since it might assume there is a problem where there is not. How well it fits the operation and maintenance industry on which this paper focuses is at this point open for discussion and will be

explored in a later section. It is mainly a question of whether opportunism is actively present in the industry.

Most of the issues regarding contracting theory are rooted in assumptions the theory makes. For example, contracting theory assumes information symmetry, which is unlikely to be the case in the real world. Although there are discussions within contracting theory, generally the Principal-Agent theory is used as a complementary theory in order to work with information asymmetry and discuss incentives.

Overall, as I have touched upon in sections 3.1 and 3.2, the theories are useful for analysing blockchain technology and smart contracts. Transaction Cost Economics is applicable to blockchain technology, where it emphasises the importance of decreasing opportunism. Contracting Theory is applicable to smart contracts, where it can be used to identify how well smart contracts deal with opportunism. This is linked to whether a smart contract can be considered a complete or an incomplete contract, where in the case of a complete contract, it will theoretically decrease opportunism. While in the case of an incomplete contract, it will not decrease opportunism.

4. Smart contracts

This section, as the title suggests, will be about smart contracts. First, the properties and mechanics of smart contracts will be discussed. Then, the definition of a 'smart contract' is discussed. As will be made evident, there is still a significant discussion going on in this area, and each author has his/her own definition and viewpoint. Therefore it is important to properly discuss this topic and, based on common aspects of a definition between different authors, to give my own definition of a smart contract. The third part of this section will discuss smart contracts on Ethereum. The fourth part of this section will describe the advantages and disadvantages of smart contracts on a blockchain network. The fifth, and final, section will summarise the findings in this chapter.

The goal of this section is to answer the question: to what extent can smart contracts using Ethereum be defined in contracting theory? This will be done by testing the assumptions behind complete and incomplete contracts to assumptions behind smart contracts. Apart from answering this question, it is also necessary to address how smart contracts deal with opportunism, this is done in section 4.5.1, a sub-section of the summary.

4.1 Properties and mechanisms

There are several definitions of smart contracts. One of the most bare-bone definitions is by Szabo (1994), who is considered to have first proposed the term. He defines a smart contract as a computerised transaction protocol, that automatically executes the terms of the contract. Consequent definitions have made subtle differences to this definition, for example, Catalini & Gans (2017) define smart contracts as operations in more complex platforms, that require verification that a contract clause is enforced, before a certain protocol is executed.

To go even simpler than the definition by Szabo, a smart contract can be considered to be "*an agreement whose performance is automated*" (Savelyev, 2017, p. 120). Savelyev (2017) argues that by this definition, even a vending machine for holy water of almost 2000 years ago, in which you dropped a coin and in return would get some holy water, can be considered a smart contract, after all, the process is completely automated. The definition by Szabo, then, shows a more modern definition, since it mentions that a smart contract is a computerised protocol. This puts it already in more modern terms, however, it is still a very broad definition, and current vending machines have more computerisation behind them. Is there something in the definition of smart contracts that will make it more than just an abstract description of a vending machine?

There is very little discussion in the literature regarding how a smart contract operates. A total of six properties can be identified (Christidis & Devetsikiotis, 2016; Lauslahti, Mattila, & Seppälä, 2017; Luu, Chu, Olickel, Saxena, & Hobor, 2016; Savelyev, 2017; Tjong Tjin Tai, 2017). The first property is that the contract is completely self-enforcing, it gets some input that triggers its activation, and then automatically executes what its rules state that has to happen. The second property is that the contract is completely electronic, it can *only* exist in an electronic form. The third property is that the contract rules and agreement consist of lines of code, which makes the smart contract something like a software program. A fourth property which follows from this is that a smart contract cannot be vague, software coding does not allow for multiple interpretations of a rule. This makes smart contracts a very clear set of rules, of which no rules can be missing for the execution of the contract, all probabilities have to be accounted for. However, it also makes the smart contract vulnerable to any problems and security issues that the software it is built on is vulnerable to. A fifth property is that a smart contract has a conditional nature, since conditions are inherent to software coding, where rules are likely to be in the form of: *if (x) do (y),* or *when (x) happens, check for (y)*. The contract thus cannot produce an output different than (y) for input (x), in other words, the contract is deterministic. The sixth and final property is that the contracts are self-sufficient, this term has a slightly different emphasis compared to the self-enforcing principle. Self-sufficient refers to the fact that smart contracts do not need legal institutions or legal enforcement agencies in order to exist, because it does not use them to execute its rules. Another way

to say this is that smart contracts allow for transactions in a *trustless* network, since there is an absence of a trusted intermediary, and you do not have to trust your transaction partner while contracting with them (Savelyev, 2017).

To summarise the properties discussed above, smart contracts are:

(1) Self-enforcing,
(2) Exist only in an electronic format,
(3) Software-based,
(4) Clear and unambiguous wording of its rules,
(5) Conditional in nature, and
(6) Self-sufficient, or trustless.

These principles reflect how smart contracts work. Now the difficulty is to translate these principles into a definition fitting into contracting theory and Transaction Cost Economics.

4.2 The definition 'smart contracts'

Clack, Bakshi, and Braine (2016) state that the term smart contract is generally used in two different ways: smart contract code, and smart legal contracts. Smart contract code refers to software agents that fulfil certain obligations and exercise certain rights, and may take control of certain digital assets within a shared ledger. Smart legal contracts focus on how legal contracts can be expressed and implemented in software. In the case of business-to-business contracting, a smart contract is used in both forms: the contract attempts to represent a legal contract, with the capabilities of a smart contract code.

Aside from the two different ways of using the term smart contract, there can also be considered to be two types of smart contracts (Governatori et al., 2018; Levi & Lipton, 2018). There are "code-only smart contracts", and "ancillary smart contracts." Code-only smart contracts are contracts that solely exist as executable code on software. Ancillary smart contracts are smart contracts that execute certain clauses of a traditional text-based contract, in which is also referred to what actions are done by the smart contract. It supports a traditional text-based contract. In my opinion, the type of smart contract used will depend on the needs of the contracts. For example, a contract can be too vulnerable to possible legislative changes to be completely on software. Instead, the text-based aspect of an ancillary smart contract ensures that possible changes can be made in case it is required. The six principles behind smart contracts as discussed above will be the same for both code-only and ancillary smart contracts. However, when analysing the two types of smart contracts from the perspective of the abstract theories as discussed in this paper, they are different.

When using contracting theory for analysing what a smart contract is, the first question that needs to be answered is whether a smart contract is a complete, or an incomplete contract. The self-enforcing and self-sufficient character of a smart contract points to a complete contract. However, in the case of ancillary smart contracts, these are clearly incomplete contracts by themselves. They are inherently incomplete because they exist due to an assumption that the contract cannot be made complete on a code-only base, partly due to expected changes in for example legal circumstances, of which the exact form cannot be accurately predicted/determined at the time of contracting. On the other hand, code-only contracts are inherently complete contracts. For example, if something happens and the contract has no rule in it that tells it what has to happen, nothing will happen, and things can go very wrong (also because there is nothing to fall back to). Therefore, code-only smart contracts have to be complete in order to function, unlike ancillary smart contracts who can fall back to an adaptable traditional contract that operates parallel to it for situations it cannot deal with. Overall, the general definition I give to a smart contract is "*a computerised contract in an online environment, that executes its protocol automatically when triggered by input from either an outside or an inside source, without needing an enforcing intermediary*."

4.2.1 Code-only smart contracts

In this section the focus will be on describing code-only smart contracts in the same terms as contracts in contracting theory. Due to a lack of scientific literature in this area, I was forced to draw conclusions based on the information on smart contracts (their properties, definitions, interpretations) as has been described in the sections above.

Code-only smart contracts are inherently complete. This means that the parties have to behave rational (missing outcomes would mean that the contract is incomplete). The conditional nature of the smart contract does not allow vague or ambiguous rules and statements, specific actions have to be done. This also excludes statements such as 'party X has to put in enough effort to reach goal Y, before he receives payment Z', in the case of a code-only smart contract, there have to be tangible and measurable things that party X has to do to reach goal Y, that represent the effort party X put into reaching the goal. Therefore, the state should always be observable by a third party, and is thus always verifiable. Because the contract is complete, both parties have to agree on the contract's final version. Therefore, the parties will focus on maximising the surplus (which represents optimal use of the relationship). However, personal profit and the relation between investment and payout is important for investment incentives. Thus, while the parties focus on maximising the surplus, they also try and maximise their personal profit. Also, because the contract is complete and self-executing, and self-sufficient, there is no need for the parties to renegotiate. The final assumption of theoretical complete contracts is information symmetry. While making the contract, the information is likely to be largely symmetric: this is necessary in order to write a complete contract. However, once the parties start executing the contract, there can form information asymmetry. It can also be argued that information asymmetry can be present from the very beginning, however, withholding information form the other party can only be beneficial if it can be exploited for extra profit. This may mean that the party can say its investment or production costs are higher than expected, taking the difference as extra profit. However, the contract can contain controlling mechanisms that prevents this from happening. And any exploitation that requires renegotiation to be useful will not be useful, because the contract is considered complete and thus renegotiation is not necessary. Any information asymmetry after the start of the project is unlikely to be of influence. Because the contract is complete, it should always, in any state of the world, specify what has to happen. Also, what is useful information for one party, might not be interesting for the other. For example, the electrician would find it useful to know what the exact voltage is on the wiring in the house to check whether it fits the standard, while the owners are only interested in whether their electronics will function when connected to the network. Thus depending on the shape of information asymmetry (either due to lack of interest or due to malicious intent), it has, or has not, an impact on the final product.

To summarise the assumptions behind code-only smart contracts:

(1) Information symmetry only in the beginning,
(2) The parties behave rational,
(3) State is always observable after the transaction, and verifiable,
(4) Companies try to maximise their own profit, while maximising surplus, and
(5) There is no need for renegotiation.

Table 3 – Comparison assumptions complete contracts, incomplete contracts, and code-only smart contracts

	Complete contract	Incomplete contract	Code-only smart contract
Type of contract	Complete	Incomplete	Complete
Information symmetry	Yes	Yes	Only during contracting
Information asymmetry	No	No	After contracting

Parties behave rational	Yes	Yes, but outcomes may not be describable	Yes
State is always observable, and verifiable	Yes	To some extent	Yes
Maximise own profit, while maximising surplus	Yes	Yes	Yes
There is need for renegotiation	No	Yes	No

Table 3 shows a comparison of the assumptions behind complete contracts, incomplete contracts, and code-only smart contracts. In this table, it is observed that code-only smart contracts fit all assumptions of complete contracts except for information symmetry, which is only present at the time of contracting, instead of during the entire project. Therefore, code-only smart contracts are defined as complete contracts in terms of contracting literature.

4.2.2 Ancillary smart contracts

In this section the focus will be on describing ancillary smart contracts in the same terms as contracts in contracting theory. Due to a lack of scientific literature in this area, I was forced to draw conclusions based on the information on smart contracts (their properties, definitions, interpretations) as has been described in the sections above.

Ancillary smart contracts are inherently incomplete. For ancillary smart contracts, the parties can also behave rational. However, describability of all the different outcomes can be impossible. Therefore, while the parties can foresee and reason all the outcomes, they are not described in the contract. The back-up by the paper counterpart of the contract ensures that a solution can always be found for any problem to which the necessary obligations are not described in either the smart or the paper contract. The parts of the contract on the smart contract can always be observed by a third party and verified. From the paper counterpart there may be some fewer observable aspects, however, mechanisms can be build-in to ensure that anything is always to some extent observable (and thus verifiable). Thus, the ancillary smart contract itself only contains rules that can always be observed by a third party and are verifiable.

While the rules of the ancillary smart contract are not complete, it is still important that both parties agree to the things it does. Therefore, maximising the surplus (insofar as that is possible to record in the statements of the contract) will be a goal for both parties. And, just like with code-only smart contracts, personal profit will also be important for investment incentives. Finally, because the contract is inherently incomplete, and assumes that unexpected or undescribed problems will occur, there will always be a need for renegotiation. However, the ancillary smart contract itself is unlikely to change, only the paper counterpart it supports. Information asymmetry is thus also more likely to be present during all stages of the project (compared to code-only smart contracts), because renegotiation can be exploited for gains from this. Even in the ancillary smart contract itself there can be information asymmetry, after all, it is only a support, or extension, from a paper contract, which still has all the vulnerabilities.

To summarise the assumptions behind ancillary smart contracts:

(1) Information asymmetry,
(2) The parties behave rational, but the outcomes may not be describable
(3) The state can be completely observed, and always be verified,
(4) Companies try to maximise their own profit, while maximising surplus, and
(5) There is always renegotiation.

Table 4 shows a comparison of the assumptions behind complete contracts, incomplete contracts, and ancillary smart contracts. It shows that ancillary smart contracts fit nearly all assumptions behind incomplete contracts. The only difference is that state can always be completely observed and verified, instead of always to some extent (but not completely) observed, and always verifiable. Also, it is more likely for there to be information asymmetry, instead of information symmetry. Therefore, ancillary smart contracts are defined as incomplete contracts in terms of contracting theory.

Table 4 – Comparison assumptions complete contracts, incomplete contracts, and ancillary smart contracts

	Complete contract	Incomplete contract	Ancillary smart contract
Type of contract	Complete	Incomplete	Incomplete
Information symmetry	Yes	Yes	No
Information asymmetry	No	No	Yes
Parties behave rational	Yes	Yes, but outcomes may not be describable	Yes, but outcomes may not be describable
State is always observable, and verifiable	Yes	To some extent	Yes
Maximise own profit, while maximising surplus	Yes	Yes	Yes
There is need for renegotiation	No	Yes	No

4.3 Ethereum

In the introduction, blockchain technology was already discussed to a certain extent, with appendix C further detailing the functioning of blockchain technology. Here will be explored how Ethereum functions, and what it adds to earlier blockchain applications, particularly bitcoin. Then, I will elaborate on why I chose to use Ethereum as technological base to do my analysis regarding smart contracts on blockchain technology.

Ethereum is a public, permissionless blockchain. This means that the data on it can be observed by anyone, even people outside the network, and anyone can join the network (there are no prerequisites to joining). The main advantage Ethereum offers over other (older) public permissionless blockchains is its use of a Turing-complete programming language (Buterin, 2013; Luu et al., 2016). Turing-complete means that a machine can do any calculation as long as it is within the limit of its memories. This is linked to the halting problem in programming, which states that it is difficult determining whether a program will stop (i.e. halt) or run forever, based on a description of the program and some input. In a distributed server you do not want a program that runs forever, since it would slow down the entire system. Bitcoin programming language is not Turing-complete, because it simply does not allow loops and such to be programmed on it. Thus, it sacrifices the ability to write complex protocols on it, to prevent the halting problem. Ethereum does want to allow complex protocols (it calls these 'smart contracts'), so it had to find a different way of dealing with the halting problem.

To prevent the system from slowing down by having these complex protocols run forever, Ethereum uses what it calls *gas* (Bhargavan et al., 2016; Buterin, 2013; Levi & Lipton, 2018; Luu et al., 2016). It works similar to how the amount of gasoline limits the range of a car. Imagine two users on a platform. One user is an external user (a person), the other user is a complex protocol running on the network. When the external user wants the protocol to do something, he has to send it a message with information. An amount of gas is sent alongside this message: a certain amount of gas is taken to reach the protocol, then gas is consumed by the protocol to execute itself. Gas that remains after the script is done running is returned to the external user. Transaction fees are taken by miners as compensation for registering the transfer of the gas, both to and from. This fee is in the form of gas as well and is simply subtracted

from what is send. Because the amount of gas consumed and paid for transaction fees is dependent on the size of the message send, the larger the message, the more gas is necessary. And the more work the protocol has to do, the more gas it needs to successfully finish its execution. One can say that a complex protocol on Ethereum will be carried out as long as enough transaction fees accompany the message. Thus, Ethereum prevents a protocol from looping endlessly and eating all the network capacity, because it stops running once it runs out of gas, regardless of whether it finished or not. Therefore, using Ethereum one can write much more complex protocols compared to blockchain languages that are not Turing-complete, because you can be certain that the protocol will at some time stop running. Even so, one can then probably argue that Ethereum is also not a Turing-complete language, but more quasi-complete.

Thus, Ethereum is much more suitable as a base for writing smart contracts than for example the bitcoin network, thanks to the Turing-completeness of its programming language.

4.3.1 Smart contracts on Ethereum

Above, it was mentioned that Ethereum describes protocols written on it as smart contracts. Vitalik Buterin, the writer of the initial Ethereum White paper at first described smart contracts as "*cryptographic 'boxes' that contain value and only unlock it if certain conditions are met*" (Buterin, 2013, p. 13). However, October 13, 2018, Buterin posted a message on twitter in which he stated that he regrets using the term 'smart contract', and instead should have gone for something like 'persistent scripts' (Buterin, 2018). The statement unleashed quite the discussion in the comments, where it became clear that the term smart contract is indeed a much-discussed term that is both hyped and unclear. The term tends to have a very broad definition, and people generally agree that it is too broad, as it makes it too simple to define an automated action as a smart contract. Instead, in the case of Ethereum it is perhaps better to describe what Buterin called a smart contract 'persistent conditional statements on a blockchain'. This definition refers to the conditional nature of what is generally considered a smart contract on blockchain technology, but also to the property of these contracts in Ethereum to be used repeatedly.

Even though Buterin may regret calling these protocols smart contracts, it is a term used widely throughout blockchain literature, often in a similar fashion to how Buterin described it. In fact, I have almost exclusively found literature on smart contracts that state it in combination with blockchain technology. It appears to be universally accepted that smart contracts are only considered to exist upon blockchain technology. Until now, smart contracts in this thesis have been described in a neutral manner, avoiding the specific link that is made between them and blockchain technology. For example, Christidis and Devetsikiotis (2016) define smart contracts as "*self-executing scripts that reside on the blockchain*." Reijers et al. (2016) use a definition that comes down to this: a smart contract is a protocol on a blockchain into which parties put assets, then the protocol redistributes the assets based on its rules.

Based on the descriptions mentioned above, a few aspects of blockchain-based smart contracts can be found already:

(1) They exist on a blockchain,
(2) They are conditional in nature, and
(3) They are self-enforcing.

Another aspect is that the transactions on a blockchain do not require a trusted intermediary, or for the contracting parties to trust each other (Kosba et al., 2016; Savelyev, 2017). This mechanism is central to blockchain technology, but it was also part of the definitions of smart contracts as identified in section 4.1. Also, because blockchain-based smart contracts are on the blockchain, and thus code, its rules are clear and unambiguous. Additionally, the contract can only exist electronic, because this is what allows it to be self-enforcing. However, stating it exists on a blockchain and is only electronic is double, therefore, I only keep the statement that it exists on a blockchain.

Thus, to summarise, blockchain-based smart contracts are:

(1) Self-enforcing,
(2) Exist only on a blockchain,
(3) Clear and unambiguous wording of its rules (formal semantics),
(4) Conditional in nature, and
(5) Self-sufficient, or trustless.

On top of these aspects, there are aspects based on the principles behind smart contracts on a public permissionless blockchain like Ethereum. The first aspect is that in general, whatever is done on a blockchain is immutable (Levi & Lipton, 2018; Luu et al., 2016), it cannot be changed. This is to prevent fraudulent transactions, because changing things in one block, means that all the subsequent blocks also have to be changed, which means they have to be recalculated. However, it could be possible to adapt or change the rules as well, if the network is build to allow it. For example, the blockchain itself can store links to certain storage boxes, these links cannot be changed. However, data in the boxes can be changed. In this case, if you make an adaptable storage box(es) for smart contracts, then they can still be changed. It appears that this is possible on Ethereum (Vaibhav Saini, 2018).

The second aspect relates to the previous one. Transactions on a blockchain are final, they cannot be reversed and there is no central system to restore funds (Luu et al., 2016). Therefore, it can be difficult to get funds and data back in case of contract breach. The funds and data transferred cannot be reached by the smart contract, and if the receiver does not cooperate in returning it, then the application will not be of any help in this regard.

The third aspect is that data is transparent (Buterin, 2016; Kosba et al., 2016; Levi & Lipton, 2018; Savelyev, 2017). On the platform, anyone can see the code of the smart contract, and the content of the transactions going in and out of the smart contract. This is a consequence of how all nodes on the blockchain network need to have a consensus on what happens in the network. This is not desirable where secrecy is required, and there are current attempts aimed at solving this problem. For example, Hawk is a smart contract system that maintains transaction privacy by compiling any transaction on it into something that is only readable for parties that have the key to read it. All other parties on the network cannot read what the transaction contains. Although Hawk tackles the transaction privacy issue, it does not tackle the smart contract code itself. You can publish only the compiled version of the smart contract onto the platform and keep the code that is understandable for people local. The contract then still operates, as it can access its own code. However, there are programs that can decompile code. The best one at the time is ‘contract-library’ (Dedaub, n.d.), which can decompile most of the smart contracts stored in Ethereum. It is not perfect, but it utilises past experience with for example variable names to make estimations and improve accuracy, even if variable names are omitted in a compiled version of a smart contract.

Thus, by placing a smart contract on a blockchain, a few extra aspects are added. To summarize all of the above, blockchain-based smart contracts are:

(1) (Possibly) immutable
(2) Transactions are irreversible,
(3) Transparent,
(4) Self-enforcing,
(5) Exist on a blockchain,
(6) Clear and unambiguous wording of its rules (formal semantics),
(7) Conditional in nature, and
(8) Self-sufficient, or trustless.

4.4 Advantages and disadvantages of smart contracts

What is beneficial of smart contracts is that it can make it possible to load a very large number of low probability clauses into the contract very easily. However, this is only possible if the clauses are predefined somewhere in a database. This implies that if a firm is already busy standardising processes and predefining rules, smart contracts can add to this by automatizing standardised processes. Also, thanks to experience, the more smart contracts are made, the bigger the library of possible predefined rules can become. With the increasing digitalisation of the world (Parmar, Mackenzie, Cohn, & Gann, 2014), standardisation is something that occurs everywhere. There is a possibility that this trend will make it possible, and beneficial, to invest in the development and application of smart contract applications.

Another benefit is the decrease in enforcement costs (Savelyev, 2017). Because the contract is self-enforcing, the only moment enforcement by a third party is necessary is when the contract is inadequately programmed to deal with the problem. From a theoretical perspective, this is not possible for a code-only smart contract, since it is supposed to be a complete contract. Ignoring the possibility that code-only smart contracts still require enforcement by a third party from time to time, ancillary smart contracts that are incomplete will still need enforcement like traditional text-based contracts, since it is only an extension of one. However, enforcement costs can decrease with the use of ancillary smart contracts and are certain to decrease for code-only smart contracts.

The self-enforcing aspect of smart contracts reduces the workload of employees in the areas in which the smart contract operates. This can leave them more time for other tasks. This can increase the overall productivity of the workforce (Dai, Mahi, Earls, & Norta, 2017; Savelyev, 2017).

A final benefit of blockchain-based smart contracts is that the contract is difficult to breach (Governatori et al., 2018). However, if the contract has a text-based counterpart, then breaching the contract as a whole is approximately as easy as breaching a text-based contract is.

4.4.1 Disadvantages blockchain-based smart contracts

There are a few issues with smart contracts on a blockchain. The first issue with smart contracts is how they are written. Because they are a code, a technical expert is necessary to translate the contractual agreements into the lines of code that form the smart contract (Frantz & Nowostawski, 2016; Levi & Lipton, 2018). For a non-technical company, these contracts would thus need to be written by an outside expert. However, then the problem is whether the companies will still be able to understand the contract, or will they continuously need contact with the outside programmer to read it? Reading the contract can probably be made possible using some translation template that expresses the code in normal language. Especially simple statements like: if the product is delivered at <party A>, <party A> pays <amount X> to <party B>, the things between <> are then parameters that have been filled in by the parties. However, another question is whether the code adequately does what it is supposed to do (is it properly programmed?). This can only be verified by a technical expert, assuming he will observe the mistake made. Also, a smart contract's terminology will differ from a text-based contract: some things left implicate in text-based contracts are detailed in smart contracts, and some aspects mentioned in text-based contracts are not written in smart contracts (e.g. because they do not need to be automated) (Governatori et al., 2018). Thus, especially code-only smart contracts will be difficult to use for companies. The textual counterpart of the ancillary smart contract makes it easier to understand and use for companies. Frantz and Nowostawski (2016) propose a solution in the form of a program that translates human-readable text into code for a smart contract. While this does not make the code more understandable for non-expert people, it does allow a wider audience to create smart contracts, omitting the absolute need for a programming expert. While this approach might be subject to errors in the case of more complex contracts, simple contracts should be doable.

The second issue is that blockchain cannot be used to store large amounts of data (Duchenne, 2018; Governatori et al., 2018). While there is no specific limit to the types of data it can store, size of the files is a hindrance. This is because generally all users are needed to keep the system running. Large files then significantly slow down the network and can make it more difficult for people to use the network. For example, in the case of Ethereum, to use it one has to download the entire blockchain, and with each block the chain becomes bigger. Not everybody will be able to spare over 20 gigabytes of space on his/her computer in order to use the network. While there might not be a need to store large amounts of data on the network when it is used for contracting, it does limit the potential applications of it within firms.

4.4.1.1 Legal issues

A third issue is concerning the legal stance of blockchain-based smart contracts (Kshetri, 2017; Lauslahti et al., 2017; Levi & Lipton, 2018; Roubini, 2018; Savelyev, 2017). Since there are no official institutions working as an intermediary, or backing the contract, it can be unclear in how far the agreements in the smart contract are legal. Specifically, in the case of legal rights when things go awry. For example, if your online bitcoin wallet gets hacked and your bitcoins stolen, then you have no chance to get them back. There is no legal institution or rights that you can turn to for compensation or help, you lost them, and will never get them back. Also, governments are still lacking any regulations regarding smart contracts (or blockchain technologies in general), which can pose a problem if a legal framework gets introduced that does not align with how the technology is used (for e.g. smart contracts). Finally, a smart contract does not care under what kind of circumstances it has been written. Whether person A was threatened by person B to transfer his bitcoins, or whether this was done voluntarily, are two very different situations from the view of standard law, but nor for the smart contract (Savelyev, 2017).

On the other hand, the legal enforceability of smart contracts is likely not an issue (Levi & Lipton, 2018). Several papers discuss the legal stance of blockchain-based smart contracts (see for example Savelyev (2017), Lauslahti et al. (2017), Governatori et al. (2018), and Levi and Lipton (2018)). The main conclusion is that while laws specifically targeting smart contracts are not (yet) present, in general a smart contract is a contract in a legal sense and is binding accordingly. However, it is also acknowledged that the continued digitalisation of the world might pose problems in future digitalisation of contracts, and thus it is important that the law in this area is reconsidered in the near future.

Nevertheless, there is the problem that there can be two situations at the same time (Savelyev, 2017). To go back to the earlier situation: person A claims that person B threatened him to transfer a certain amount of digital currency. The immutability and irreversibility of blockchain technology would still consider person B to be the owner of the transferred digital currency, while the legal institutions consider person A to be the rightful owner. There is currently no reliable or agreed-on method to deal with these types of situations.

There are two final legal issues. According to law, it has to be possible to terminate or modify a contract under certain circumstances (Governatori et al., 2018). For example, renegotiation has led to a need for change. However, blockchain-based smart contracts are potentially immutable, depending on the network. Thus, you would have to rely on perfect programming of the contract, so as to prevent the need for early termination or changing. This is unlikely to be the case, even if one only considers bounded rationality to be a factor. Termination should be possible if such a function is build-in. Changing the contract, however, is not possible. One can change the data inside the contract, but not the rules. Therefore, this last part is the main hurdle when it comes to blockchain-based contracts.

4.4.1.2 Security issues

The sixth issue is privacy (Kosba et al., 2016; Li et al., 2017). Any blockchain network on which the smart contract runs will be transparent, meaning that all users on the network can observe the history. For a company, it is undesirable for information to be available outside of the intended stakeholders.

So, if the company sets up a network in which it will regulate smart contracts with contractors and clients, it will likely desire some form of added privacy so parties outside of the intended contract, but on the network, cannot observe progress made in the contract (or the contract itself, for that matter).

The seventh issue is what can be referred to as the 'inconsistent trinity' (Roubini, 2018). This trinity consist of security, decentralisation, and scalability. The main point is that any attempt to improve on any one of the three aspects leads to a decrease in performance of at least one of the other two. Currently, scalability is sacrificed for security and decentralisation, as is observed in the low throughput of the Ethereum system. This trade-off can be mainly attributed to what can be referred to as a "51%-attack." The technology ensures consensus through computation (Kshetri, 2017; Roubini, 2018). Thus, if you want to forge a transaction, you will have to out-compute the rest of the miners on the blockchain. This is possible if you manage to obtain control over at least 51% of the computational power of the blockchain. Thus, if the chain is decentralised, it becomes more difficult to obtain the majority of the computational power. However, the scalability of the platform will also decrease because a significant larger amount of people is actively computing on it, compared to more centralised consensus mechanisms. The main alternative to the Proof of Work consensus mechanism is Proof of Stake, which is discussed in appendix C.

The eighth issue is that Ethereum suffers from security issues due to weaknesses in the code (Dai et al., 2017; Roubini, 2018). A solution to a security leak, or a problem found in the Ethereum code, could be to hard fork the blockchain. This means that the entire system gets something like a 24-hour reset and continues using new rules. However, the old version remains in existence. Thus, two versions of the network then exist, each operating on different rules. Users can decide whether they want to remain on the original network, or want to move to the new one (Wong & Kar, 2016). However, this hard fork is a serious problem for business applications of smart contracts. What do you do if contracts that are supposed to be immutable, inviolable, can suddenly be overturned via a majority vote? What guarantees the security and reliability of your network? A potential solution is for the business to set up its own private blockchain network, instead of working on a public blockchain like Ethereum.

The ninth and final issue is that when the blockchain platform does not use formal semantics in its programming, this can lead to security issues (Dai et al., 2017). Formal semantics refers to the use of language that cannot have multiple interpretations. Earlier was stated that smart contracts exist of clear, unambiguous statements. While this is theoretically the case, and likely due to the fact that smart contracts are programmed, the possibility remains in actual applications.

4.5 Summary and discussion

Thus code-only smart contracts are complete contracts, and ancillary smart contracts are incomplete contracts. Both fit the assumptions and descriptions of complete and incomplete contracts as defined by the theory fairly well. For dealing with opportunism, ancillary contracts do not need any changes compared to the old system which uses text-based contracts. This is mainly because ancillary smart contracts are incomplete, and a text-based counterpart is present to incorporate any changes. Thus, this type of smart contract helps in reducing some costs, like enforcement costs, and increase time efficiency in the areas it is applied, but will not change how the market already deals with opportunism. On the other hand, code-only smart contracts decrease renegotiation costs and increase efficiency and enforcing of a contract. From a theoretical perspective, complete contracts are best for decreasing opportunism, because they limit the sources of opportunism (e.g. there can be no discussion as to what to do in an unforeseen situation). Add the capabilities of smart contracts to this, like how it is impossible to have multiple interpretations of a statement, it will have a significant impact on opportunism.

Aspects of blockchain-based smart contracts are:

(1) (Possibly) immutable,
(2) Transactions are irreversible,

(3) Transparent,
(4) Self-enforcing,
(5) Exist on a blockchain,
(6) Formal semantics,
(7) Conditional in nature, and
(8) Self-sufficient, or trustless.

Benefits of smart contracts are that they improve the efficiency of the contracting processes by automation of certain processes, they are difficult to breach, it becomes easier to describe a large number of low probability cases, and transaction costs decrease (in this case the costs associated with writing and enforcing the contract).

However, there are also a number of issues with blockchain-based (Ethereum) smart contracts:

(1) Difficult to read and write for non-experts,
(2) Cannot store large amounts of data,
(3) Uncertain legal stance, although generally enforceable,
(4) Cannot be changed,
(5) Has to be terminated,
(6) Privacy,
(7) 'Inconsistent trinity',
(8) Ethereum has weaknesses in the code, and
(9) There is the possibility of not using formal semantics.

The issues that will mainly pose a problem for the adoption of smart contracts for business-to-business contracting will be the uncertain legal stance of the smart contracts, that they (possibly) cannot be changed, the 'inconsistent trinity', privacy, and possible weaknesses in the code (Governatori et al., 2018; Wong & Kar, 2016).

Unfortunately, that code-only smart contracts are complete contracts has quite the consequence for applying them in the business environment. In section 3 it was discussed that firms use incomplete contracts in transactions to deal with opportunisincm. Fitting complete contracts into this network of incomplete contracts will not be easy. Even if you consider that complete contracts are supposed to decrease the risk of opportunism, transaction costs like writing the contract prevent the successful creation of complete contracts. In this regard it makes no difference whether these contracts are in a digital environment or on paper.

Another question that has to be raised is whether code-only smart contracts are actually possible, ignoring their requirement that they only need to be (obligationally) complete. Text-based contracts have two aspects: operational, and non-operational (Clack et al., 2016). Operational aspects are the aspects that can be automated, and thus be placed in a smart contract. Non-operational aspects are those aspects in the contract that cannot be (or are not wished to be) automated. In the case of ancillary smart contracts, both aspects are still present in the contract as a whole (consisting of the code-part and the text-based part). However, code-only smart contracts stand on their own, which means that the non-operational aspects of a contract are missing in these. Because the non-operational aspects can still be very complicated, even for a simple contract (Clack et al., 2016), it is unlikely that these can be completely removed from a contract. This supports the point made in the previous paragraph, leading me to conclude that it is likely that only ancillary smart contracts will be used by firms. However, I will continue to consider both types an option for completeness of analysis.

4.5.1 Smart contracts and opportunism

Blockchain technology is a technology that is built on trust, or the lack thereof between potential transaction partners. It deals in a very different manner from opportunism, where you generally know your transaction partner, allowing you to take repercussions in case that is necessary. So, how would

smart contracts decrease opportunism, and its associated transaction costs? In this section, this question is explored, using information on smart contracts (their properties, definitions, interpretations) and Transaction Cost Economics as has been described in earlier sections and chapters.

Transaction Cost Economics proposes that complete contracts are ideal in that opportunism is not possible under such a contract. However, it also proposes that complete contracts are difficult to write due to transaction costs. Assuming writing a complete contract in the form of a code-only smart contract becomes possible due to digitalisation and standardisation, another question comes to mind: do code-only smart contracts deal as effective with opportunism as a complete contract is suggested to do in contracting theory? While a smart contract does not leave room for multiple interpretations, it is self-enforcing, which means that the trustworthiness and accuracy of the input is important. However, unless build in the contract, there is no one who checks the trustworthiness (and accuracy) of the input. In other words: how to control the link between the offline object and its online representation in the contract?

To illustrate: imagine that the installation of a radiator in a new home can cost a maximum of €2,000, as is stated in the contract. Then who will prevent the installer from saying it cost €2,000, while it was actually €1,800? The installer abuses the space the contract leaves. In other words, the contract is subject to opportunism. One can argue that in this case the contract just was not precise enough, and that the rule should state: install a radiator of type X, if it is done, receive €1,800 (assuming that the installation of type X will cost exactly €1,800, nothing more, nothing less). But in this case, there is the problem that there is no one to control whether the radiator of type X is actually installed! If a cheaper type is installed, the installer can still claim to have installed type X and run with any leftover money. Or worse, the installer claims to have installed it while he has not. Introducing extra control on the implementation of the contract defeats the self-enforcing purpose of the contract, then you might as well not do the contracting via a smart contract. However, if control is already something that normally occurs, then the automatization that the contract provides can still decrease overall costs, since the control on whether it was actually done is normal and is not considered extra costs. However, this does not solve the issue of opportunism.

The main issue that can prevent code-only smart contracts from efficiently decreasing opportunism, is that it is an online entity, often describing offline actions and products. The link between the offline entity and its online representation in the contract is crucial, and anything that is missing can impact the trustworthiness and operation of the contract. While in an ideal world this would not be a problem (everybody would be honest and update accordingly), such levels of trust are lacking in the actual environment (which is the whole reason opportunism needs to be decreased). Fortunately, smart contracts cannot be breached. It should be fairly simple to build-in standard rules into smart contracts that prevent abuse of the input. For example, once it receives input from certain sources, say one of the contracting parties and a third unrelated party that is designated to be trustworthy, that the other party has breached the contract, then the smart contract can set in motion some countermeasures. It could be allowed to reverse the money given, or it could automatically start a court case, until both parties give the input that the issue is resolved. These types of issues are quite standard, therefore making standard rules and regulations for dealing with these that are present in every smart contract created on the system, should be possible, and beneficial. To conclude, code-only smart contracts can significantly decrease opportunism, provided that the mechanisms to do so are build-in.

Ancillary smart contracts are easier in this regard. They are incomplete, and have a text-based counterpart to deal with any unforeseen issues, or necessary changes. Therefore, when they are applied, ways to control opportunism that are already used in the industry are likely to remain. It could decrease some opportunism by removing discussion on the interpretation of the operational aspects of the contract, and by adding some more, often forgotten or detailed clauses that can now be easily taken out of a database and added. It could potentially also get some similar system like what I discussed for code-only smart contracts. Especially when the two types of smart contracts are used simultaneously

on the network, this would be easy to add. Then, it does decrease opportunism to some extent, although to a lesser extent than code-only smart contracts, because of the non-operational aspects on the text-based counterpart that can still create discussion. As well as the need to discuss what to do in unforeseen situations.

5. Infrastructure division of the operation and maintenance industry

This section has been written with the help of R. van der Krol (personal communication, October 4, 2018), C. Beltman (personal communication, October 1, 2018), P. Dourlein (personal communications, November 2, 2018), F. van Ruth (personal communication, October 24, 2018; November 7, 2018), J. van der Gaag (personal communication, October 26, 2018), R. Grosscurt (personal communication, November 5, 2018), R. Snijders (personal communication, October, 31, 2018), and M. Hartsema (personal communication, October 23, 2018). Any statements in the entirety of section 5, that contain no academic reference have been made based on interview findings. This section attempts to answer the sub-question: "to what extent can technical contracts and performance contracts be defined in terms of contracting theory?"

To start with, why the focus on this industry? This industry division is behind in terms of digitalisation and standardisation, but trying to catch up. Therefore, there is a significant amount of innovation in the processes and in information technology. Blockchain technology and smart contracts are especially influential in the field of information technology, making it a valid option to explore when innovating in this area.

Another important reason is something that is inherent to this particular industry: opportunism. Section 5.3 elaborates on this further, but to summarise: opportunism is a prominent issue in the industry and has a significant influence on how each party acts. Since Transaction Cost Economics focuses on how to deal with opportunism, and for business-to-business transactions (as is the case in this industry) the operation and maintenance industry is exceptionally fitting to analyse, considering the theoretical lens that this paper uses.

There are three levels when it comes to assets in the operation and maintenance industry: there are the asset owners, the asset managers, and the service providers. Contracts in the industry are generally between the asset managers and service providers.

Figure 4 shows a drawing of the industry when looking from the perspective of contracts. This is merely an example, and is neither accurate nor universally applicable. However, it suffices for demonstration purposes. In the top layer there are the asset managers, they put out requests for projects. At the bottom layer, there are the service providers, they do the physical work that needs to be done to fulfil the contract. In between the top and bottom layers, there is the middle layer. Parties in this layer can operate as both an asset manager and a service provider. Arcadis can be located in this layer.

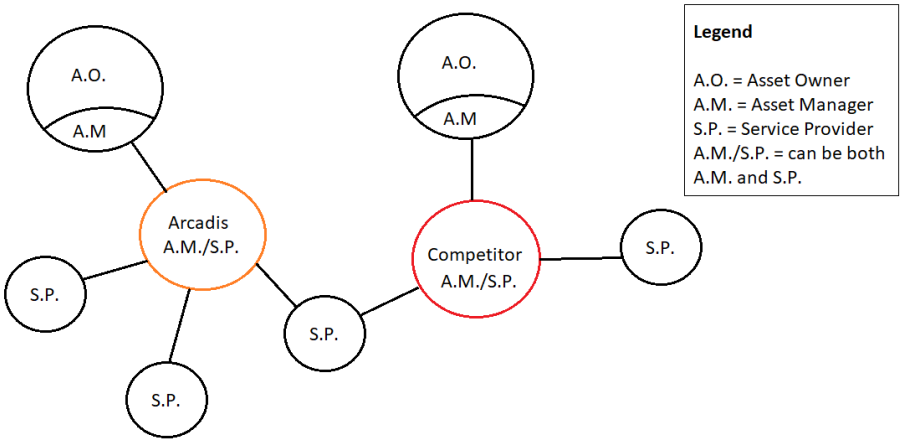


Figure 4 – A possible ecosystem

There are two extremes in contracts in the operation and maintenance industry: there are technical contracts, and performance contracts. Technical contracts are relatively short term and have clear goals and actions that are required. On the other hand, performance contracts span across a long period of time and contain some goals, but the goals lack definition. In practice, there is a vast range of contracts between these two types, however, to simplify analysis I only analyse the extremes. In the next sections I will further explore the two extremes and attempt to express them in contracting theory terms.

5.1 Technical contracts

Technical contracts are precise in what they want, you can also say that they are obligationally complete contracts with unambiguous statements: there is no discussion as to what has to be done, even if not is dictated how it has to be done. This also means that the parties know what outcomes there are to the contract, in other words: the parties are rational. Because the goals are so clear and unambiguous, this also makes it simple to verify whether things have been done for a third party. Adding mechanisms like taking pictures to prove things have been done makes it even easier for third parties to observe what has been done.

Contracts start when an asset manager puts out a request, this request contains what he wants. Then service providers can react to this request and let the asset manager know for how much money they can do it. In this initial state, it is possible that either party is withholding information. For example, the service provider may have noticed something missing in the request. He can choose not to voice this. Instead, when he gets the contract, he can use this knowledge to try and get extra money to do these things (and with this increase his profit). On the other hand, the asset manager may withhold some information from the service provider. There are also different needs for information: whereas the service provider wants detailed, technical information for the project execution, the asset manager will want less specific, strategic information to know how the project is advancing. Thus, during most projects, there will be a constant difference in information between the parties, largely due to different information needs.

In practice, any contract is likely to be renegotiated, for example due to scheduling issues. Apart from that, there are also reasons such as updates on the process that dictate when the contracting parties meet again. The parties do aim to keep negotiations and meetings to a minimum. I assume that a technical contract, in its simplest form, will never need renegotiation during the process. This is because the contract form is very clear in what has to happen, and the contracts are generally reasonably short-term. Therefore, while technical contracts do not *need* renegotiation, it does generally take place.

To summarise technical contracts in terms of contracting theory, assumptions behind functional contracts are:

(1) (Obligationally) complete,
(2) The parties act rational,
(3) State is observable and verifiable,
(4) There is no need for renegotiation, but it will generally take place
(5) Information asymmetry,
(6) Companies try to maximise their own profit, while maximising surplus.

5.2 Performance contracts

Performance contracts state goals of what has to be done, without specifically defining how the goal should be reached (e.g. not prescribing methods on how to keep the grass short enough) (Vraalsen, Aydt, Mendes, & Reed, 2007). There can, however, be more technical sub-goals listed in them, such as safety requirements. This contract type leaves significant space for the service provider to tackle any problems and find the best way for him to reach the goals based on his personal strengths and weaknesses. A typical performance contract spans over several years, which means that there is always a need for renegotiation at some point. For example, if some regulations change, it can force the parties

to re-evaluate the contract and adapt some aspects, like extra funds for the service provider to add additional safety measures. This also implies that the outcomes of the contract can be indescribable at the time of contracting: when you have a performance contract to design a bridge, build, maintain, and to take it down again, then there are unknowns like how the bridge should look like at the time of contracting.

Because the contract leaves significant room for the service provider, there is also significant information asymmetry. The service provider can use information he has, but the asset manager not, to increase his bargaining position during renegotiations. The same works the other way around. There is also, just like in technical contracts, a difference in the type of information that is desired by each party. A performance contract can be between an asset owner and an asset manager/service provider, but also between an asset manager and a (or multiple) service provider. For these types of contracts, the service provider gets a certain amount of money in set intervals (e.g. every month) to reach the goals in the contract. Whatever he saves in costs, he can keep for himself. Thus, it is in the best interest for the service provider to be as efficient as possible in reaching the contract goals, so he maximises his profit. Because a performance contract is over a long period of time, it is beneficial for the contracting parties to focus on maximising the surplus, aside from their personal profit. This will ensure a healthier relationship and prevent (court) issues in later periods of the project.

Because the contract clauses can be vague and ambiguous, there can be an issue with observability of the progress for a third party. For example, how can a court judge whether the service provider put in enough effort in a certain area (because how do you quantify effort?)? There are a few mechanics in place to prevent these types of problems. Companies do sample testing to see whether a service provider is delivering his job, but there are also programs that the service provider may have to use to track his progress and control his quality. Pictures are made, and progress is documented, thus there is always some proof as to what has (not) been done. Any aspect is made at least partially observable for third parties. Thus, through these types of methods, any conflict can be solved via a third party, if negotiation is no longer an option.

To summarise performance contracts in terms of contracting theory, assumptions behind performance contracts are:

(1) Incomplete,
(2) There is always renegotiation,
(3) Parties do not behave rational,
(4) Information asymmetry,
(5) Companies try to maximise their own profit, while maximising surplus, and
(6) State is to some extent observable, and always verifiable.

5.3 Transaction Cost Economics on the industry

Similar to how smart contracts were analysed in section 4, it is also important to observe the industry from a Transaction Cost Economics perspective. This perspective then poses the question: is there opportunism? If so, how is it dealt with? And where do transaction costs mainly come from?

Opportunism is definitely present, especially on the side of the service providers. These companies do not show a particular interest in maintaining long-standing relationships and will do a job for anyone, including the competitor of a (current) contracting partner. Also, because of the heavy price competition on the market, they will take any opportunity to get more money out of a contract. For example, once the service provider gets the contract, he can point out that there are actions missing from the contract, that need to be done to reach the goal(s). He, in turn, can then say he will do these as well, but it will cost extra. The asset manager, at that point in time, has already signed a contract with the service provider, and is unlikely to go to another service provider. This extra work is where the service provider traditionally gets most of his profit from.

However, a more recent trend shows that service providers are less focussed on getting more profit via extra work. By listening and working together with the asset manager, there is a better foundation for a long-term relation. In addition, service providers are starting to keep a form of internal reputation system in which they record how previous jobs went with an asset manager. This helps in determining whether to take a new job and identifying what possible problems can occur, based on previous experiences.

Even so, asset managers have a number of systems in place to control what the service provider does. This is mainly because of the (currently still very present) opportunistic reputation that service providers have. To start with, the asset manager can check whether the service provider has an ISO certification. This certification means that the service provider can, using standard processes and tests, prove the quality of what he is delivering during a project. The service provider can be required to write a Project Management Plan (PMP) based on the ISO certification. The asset manager can test whether the service provider is doing the quality tests and processes adequately. The asset manager starts with a system test, which tests based on the ISO certification and the PMP. When the system test shows issues, there is the process test, which is only based on the PMP. And finally, there is the product test, which checks everything. This last test is only done when the service provider is truly lacking in ensuring his quality. The asset manager is not continuously checking the service provider, he uses sample testing to identify whether more attention is needed. This saves time, manpower, and money for the asset manager, and it is especially useful for large instances which have often multiple projects running with multiple service providers.

6. Smart contracts in the operation and maintenance industry

This section has been written with the help of M. Zanen (personal communication, October 12, 2018), M. Hartsema (personal communication, October 23, 2018), D. Stolker (personal communication, November 29, 2018), M. Zanen (personal communication, December 3, 2018), R. Opgenoort (personal communication, December 7, 2018; November 7, 2018), D.-J. van Driel (personal communication, October 18, 2018), and M. J. Grevink (personal communication, October 18, 2018). Any statements in sections 6.1 and 6.2, that contain no academic reference have been made based on interview findings.

To answer the main research question, it is not only important to compare technical and performance contracts with the code-only and ancillary smart contracts, but also with the complete and incomplete contracts as identified in the abstract theories. Table 5 provides an overview of the assumptions per type of contract and allows for comparison, it summarises what has been discussed previously in sections 3, 4 and 5. It shows that technical contracts fit best for code-only smart contracts, and performance contracts best fit ancillary smart contracts. This does not mean that technical contracts need to be complete contracts, and thus that code-only smart contracts are the only option for this type of contract. Instead, these results indicate that a technical contract is more likely to be written as a complete contract in the form of a code-only smart contract, than performance contracts. There is no guarantee that it becomes possible, but there is a higher possibility. Thus, both types of blockchain-based smart contracts have a potential application in the infrastructure division of the operation and maintenance industry. This already partly answers the sub-questions to what extent do performance and technical contracts have determinants (assumptions and operational aspects) in common with smart contracts?

Other sub-questions that will be answered in section 6 are: what are the managerial and technical consequences of applying smart contracts using Ethereum to technical contracts and performance contracts in the infrastructure division of the operation and maintenance industry? And finally: what problem(s) do blockchain-based smart contracts solve in the infrastructure division of the operation and maintenance industry?

Table 5 – Overview of assumptions per type of contract

	Theoretical contracts		Smart contracts		Contracts in the industry	
	Complete contract	**Incomplete contract**	**Code-only smart contract**	**Ancillary smart contract**	**Technical contract**	**Performan-ce contract**
Type of contract	Complete	Incomplete	Complete	Incomplete	Complete	Incomplete
Information symmetry	Yes	Yes	Only during contracting	No	No	No
Information asymmetry	No	No	After contracting	Yes	Yes	Yes
Parties behave rational	Yes	Yes, but outcomes may not be describable	Yes	Yes, but outcomes may not be describable	Yes	No
State is always observable, and verifiable	Yes	To some extent	Yes	Yes	To some extent	To some extent
Maximise own profit, while	Yes	Yes	Yes	Yes	Yes	Yes

maximising surplus						
There is need for renegotia-tion	No	Yes, always	No	No	No, but will generally still take place	Yes, always

6.1 What problems are there?

When it comes to blockchain technology, the first question to ask oneself is: what problem is there? The second question then is: how can we (me and other involved parties) solve this together? And finally: does blockchain technology provide the answer?

The first problem lies in the communication between a client and a contractor. When a conflict arises, you generally escalate in the layers of the companies. These layers are as follows: at the top there is the strategic layer, then the tactical layer follows, and at the bottom there is the operational layer. Figure 5 shows the two situations, S stands for strategic, T for tactical and O for operational. 5a depicts the current situation, where all layers of one party (generally the contractor), have to communicate with only a small fraction of the other party (generally the client). This is not a desirable situation, because when a conflict arises and escalation is required, higher layers on the client's side need to be involved, while the communication lines remain the same. Instead, the situation depicted in 5b is wanted, where each layer communicates with the corresponding layer of the other party.

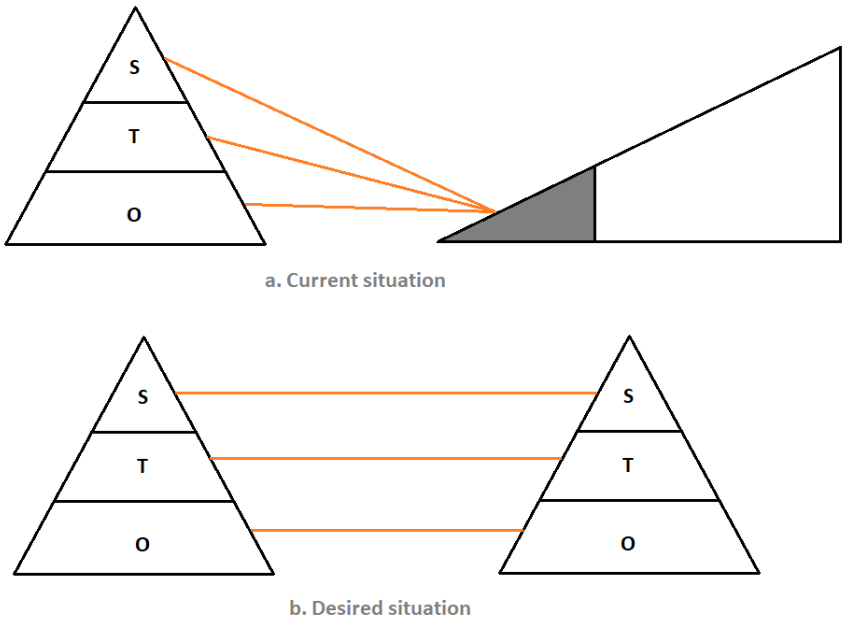


Figure 5 – Current and desired situation of communication between corporate layers

A second problem lies in the involvement of many stakeholders during a project. For anyone involved in the project it then becomes difficult to see the most recent status of the project, this can be due to a number of reasons. For example, it can be the case that each contractor has to update his part of an overarching model, send an update to a person, who then has to verify and incorporate the change(s) in the overarching model. The more contractors are involved that make updates and send them, the more updates have to be verified and incorporated. Another example is that not all contractors use the same communication software/standard, this can give some trouble in status communication between

different parties. Overall, the more parties involved, the slower the updating process and the more difficult it is to get an accurate current status overview.

A third problem is that projects are observed on an individual basis. With this I mean that each project is taken as an entity by itself, and a lack of standardisation makes it so that similar projects can have very different methods of execution. Then there is also a lack of standardisation in recording data in and between different systems, thus what is recorded in one system is not also recorded in another, and the same data can have different variable names between and within projects. This also causes a messy and dispersed data storage, which makes data retrieval on past projects difficult (sometimes just impossible). This in turn prevents analysing how a project went and correlating data, and thus to anticipate problems in current and future projects.

A fourth problem lies in data sharing. Contracting partners may not wish to share all their information, or to be transparent in what they have done exactly (e.g. when and where). This is because certain information may be vital for what the company perceives to be its competitive advantage, and sharing it can let competitors take advantage of it. Or, the other contracting party can try and dictate what one should do, based on this information. In general, the asset manager/owner has more desire for data sharing than the service provider. Another possibility lies in how data is shared. The parties agree on sharing data in a certain format, however, converting the data to the format causes data loss. This goes both ways.

A fifth problem lies in contract specifications. While it is possible that a contractor does not fulfil the specifications, it is also possible that while the contractor fulfilled the specifications, the resulting product is still not functional, or the desired product. This can be, for example, because specifications were not described properly, or (unintentionally) omitted at the time of contracting. An example of this is two speed boats that were ordered by the Dutch coastguard (Hofs & Misérus, 2018). The specifications of the boats were worded incorrect, while the boats should have been able to go through high waves on a high speed, the specifications stated that they should be able to go fast, **and** through high waves. Thus, while the boats ended up being able to do both, they could not do both things simultaneously, which is what was actually wanted.

6.1.1 Smart contract application as a solution?

The problems in the previous part can be summarised in two terms: communication and standardisation. The main benefits that blockchain provides for companies are: removal of intermediaries, network-wide consensus on the latest state, distributed data storage, and possibly transparency (depending whether the network is public or private). For contracting, the first three benefits are most likely to be useful. The last benefit can be useful depending on what the party wants and/or needs in term of data sharing to the outside world and within the network. For smart contracts, the main benefits are: they improve the efficiency of the contracting processes via automation, they are difficult to breach, it becomes easier to describe a large number of low probability cases, and transaction costs decrease. Now, how do the benefits link to the problems above? How and/or what do they solve?

6.1.1.1 Communication issues

The issues in communication are:

(1) communication between organisational layers of contracting parties in case of conflict and a need for escalation,
(2) the correct wording of contract specifications, and
(3) the need for fast data sharing in a project with many parties.

Issue (1) could potentially be solved using the smart contract application and the system-wide consensus mechanism of the blockchain network. Since all involved people can see the latest state of the project at any time, it should be easier to also connect the right organisational layers in case of conflict. To illustrate: while building a foundation, a part of the building was forgotten, and therefore lacks a

foundation. Then, the next party cannot properly build the house on top of the foundation. Instead of having to contact some person of the other organisation, the right supervisor can be contacted using the smart contract application. A function such as this might not be build-in, but it is certainly possible.

Issue (2) can potentially be solved using smart contracts. This is due to their conditional nature. While it may still go wrong due to for example bounded rationality, it is easier to confirm whether you have correctly worded your specifications. Also, if a number of specifications are standard across similar projects, then it is also easier to copy-paste these into the new contract, lessening the chance that they are written down incorrectly, or forgotten.

Blockchain technology on itself provides a solution to issue (3). Because of the consensus-mechanism (where each user always has a copy of the latest state), and the removal of intermediaries. By applying a smart contract application, any intermediary that traditionally needs to record and implement changes by all parties is removed as well (each party sends one person/party an update, that person/party incorporates all updates in one file), or the number of these intermediaries should at least reduce. Thus, the technology improves efficiency of incorporating changes via the smart contract, and the underlying blockchain network ensures that all other (involved) parties can immediately see any changes made. It is also possible to create a consensus mechanism into such a mechanic in case something needs to be changed on which multiple parties have a say, so as to prevent one party from changing any rules to his favour, or sabotaging the project.

6.1.1.2 Standardisation issues

The issues with standardisation are:

(1) project execution and data recording in and between systems, and
(2) data sharing between contracting partners.

Issue (1) could be solved by the implementation of a clear rule system, which governs both project processes and data input. In other words, standardisation of project processes and how data should be recorded and processed is necessary. The blockchain network provides a place to store data, but it is dependent on the input for any analysis. While the implementation of a blockchain-based smart contract application can solve this issue, this is only the case if it is compatible with all current systems (or just replaces them), and provides the set of rules that are needed. Therefore, neither the blockchain network or the smart contract application on it can be of any use in particular compared to other platforms in solving this issue.

Issue (2) can be solved depending on the data sharing issue. If there is a lack of data regarding the execution and process of the project shared between the contracting parties, then a blockchain-based smart contract application can help resolve this issue. By acting as a portal in which everything has to be registered (in a certain format), rules can be defined beforehand what is and is not shared. Then, either party can ensure they will always have access to data they need from the other party. A downside to this is that the parties also need to be willing to share their data to such an extent.

When the core of the problem lies in conversion of data for communication between partners, then the smart contract application can help by providing one platform trough which data is shared. However, when it comes to digital drawings or test results in certain applications, the smart contract application cannot help if the other party does not use the same program. In this case, data will still be lost in the necessary conversion(s).

Overall, I think that data sharing between contracting partners and other possibly involved parties can be facilitated only in the area of project management. Here, the smart contract application on the blockchain network can provide a way to standardise the communication regarding project progression and issues, and in this manner provide one single communication application.

6.2 Technical consequences of (potential) adoption of smart contracts

At the moment, there are already a few systems in place which partly do what smart contracts can do. Examples are VISI and SEm. VISI is a communication protocol that formally records moments upon which communication is desired in a project. It can also be used to check which/whether milestones have been reached. SEm is a Systems engineering platform, which is used for the recording of contract clauses in libraries. This digitalisation is useful for quickly finding standard contract clauses (standardisation), and thus decreases the time needed to write a contract. SEm can also automatically generate progression reports. Digitalisation and automatization provide a company with more control.

What the adoption of blockchain-based smart contracts could do in these cases, is provide a new platform to perform these actions in. Ideally, it will incorporate all the functions of the other systems, leaving only one platform to do all of it. However, if partners do not wish to join the blockchain network, the old systems will still be used. Therefore, I expect that the old systems and the new blockchain network will operate next to each other (for a while), should blockchain-based smart contracts be adopted for contracting.

Aside from this, it is important to look at how blockchain-based smart contracts operate compared to traditional text-based contracts. To recap, operational aspects of blockchain-based smart contracts are:

(1) (Possibly) immutable,
(2) Transactions are irreversible,
(3) Transparent,
(4) Self-enforcing,
(5) Exist on a blockchain,
(6) Clear and unambiguous wording of its rules,
(7) Conditional in nature, and
(8) Self-sufficient, or trustless.

These aspects need to be compared to how text-based contracts operate.

To start with, text-based contracts exist offline (and possibly online), but they are not software-based and do not exist only in an electronic format. The contract generally has a physical version that needs to be signed by both parties before it becomes active. This also means that both parties need to have some form of trust in each other, before they engage in a contractual relationship. Using renegotiations, the contract can be changed anytime. Renegotiations occur because the situation changes, or because rules were ambiguous and the different interpretations lead to conflict/confusion. This is also a consequence from the way rules and statements are written in the contract: they are not per se conditional, and can imply but not specifically mention aspects or rules (Governatori et al., 2018). Transactions do not necessarily need to be reversable (F. van Ruth, personal communication, November 7, 2018). The process with a text-based contract is also not transparent, at least not to the degree of a blockchain-based smart contract. This is specifically due to the different ways in which progress is measured. Even though there are programs that partly do what smart contracts are expected to start doing, these are not used for every text-based contract and not inherent to text-based contracts in general, including the operation and maintenance industry. Finally, text-based contracts are not self-enforcing: they merely represent the agreement between two parties. Table 6 provides an overview of the operational aspects of text-based contracts and smart contracts.

To conclude, a text-based contract operates almost in a complete opposite way of a blockchain-based smart contract. There are some slight nuances between code-only smart contracts and ancillary smart contracts. However, the ‘smart contract’ aspect of both types operates in the same manner. Considering the difference in how text-based contracts and blockchain-based smart contracts operate, the technical (operational) impact of switching from traditional text-based contracts to blockchain-based smart contracts will be significant.

Table 6 – Comparison operational aspects smart contracts and text-based contracts

operational aspect	Blockchain-based smart contract	Text-based contract
Immutable	Yes	No
Transactions are irreversible	Yes	No
Transparent	Yes	No (or at least not to the same degree)
Self-enforcing	Yes	No
Exist on a blockchain	Yes	No, exists offline (online copy on other platform can be possible)
Clear and unambiguous wording of its rules	Yes	No, wording is often ambiguous/unclear
Conditional in nature	Yes	No
Self-sufficient, or trustless	Yes	No

6.3 Managerial consequences of (potential) adoption blockchain-based smart contracts

There are a few hurdles that need to be overcome for the successful adoption of blockchain technology and the smart contract application. One of these is that contractors can be contacted on a very short notice (a few hours) to resolve an issue. If this contractor does not already work with the application, then it can be quite cumbersome for him to install it for a short, quick job. Thus there needs to be some form of registration for these jobs that are done by contractors (or contracting partners) that do not use the smart contract application.

Another issue is that blockchain is distributed, this can make the role of the different users unclear. A company wants to be certain of what its role is. It can be threatening to the company if it is uncertain as to what its responsibilities are.

An additional issue is that coercive pressures from dominant suppliers and customers are significant determinants for the adoption of a technology (Liang, Saraf, Hu, & Xue, 2007). Coercive pressures can be both formal rules and regulations, but also informal expectations on the organisation (like technical standards). For Arcadis, this means that if for example the government adopts a system similar to this, then Arcadis is also more likely to adopt the technology. Then, if Arcadis is the one to first adopt the new technology, companies and instances where it is a dominant supplier or customer to also become more likely to adopt the technology. Consequently, the more big players adopt the technology, the faster and better the diffusion of the technology. Therefore, finding the right partners to cooperate with on this platform is absolutely necessary for its diffusion.

Here the term network competence also comes into play. Firms are embedded in networks of competitive and cooperative relationships with other firms (Ritter & Gemünden, 2003) (an example of how this looks like is in Figure 4). In this, network competence refers to how well a firm manages its relationships in its network, in other words, it enables a firm to create and use relationships with other firms. Network competence consists of two components: the degree of network task execution, and the extent of network management qualifications possessed by the people handling a company's relationships. Task execution consists of tasks that are relevant for managing a single relationship, and

tasks which are necessary to manage a portfolio of relationship. There are two different types of tasks: relationship-specific tasks (establish and maintain a single relationships), and cross-relational tasks (managerial tasks widely used in general management literature). There are also two types of network management qualifications: specialist qualifications (qualifications which are necessary to handle the technical side of relationships), and social qualifications ("*extent to which a person is able to exhibit independent, prudent, and useful behavior in social settings*" (Ritter & Gemünden, 2003, p. 748)). An overview of the network competence tree is in Figure 6.

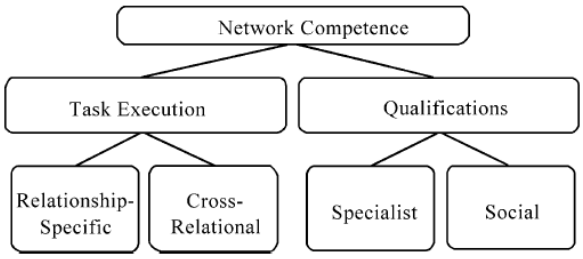


Figure 6 – Network competence tree (Ritter & Gemünden, 2003)

Related to the network competence of a firm, is the additional value of each user of a digital technology. The effect is called network effects (or network externalities) (Tiwana, 2013b). This effect refers to "*the degree to which every additional user of a platform or app makes it more valuable to every other existing user*." (Tiwana, 2013b, p. 33) The mechanic is simple: every additional user of the technology represents another user that current users can interact with. And on the other hand, the more users are on the network, the more benefit they provide to anyone joining. Thus, in the operation and maintenance industry, every additional user of the application with which you create and maintain contracts, represents another user for which the benefits of using the blockchain technology count. And for communication and speed, it is useful when each contracting partner uses the same communication and data sharing program in a project. Therefore, the more users of the technology, the more projects can be done with it and the more users will join. While a technology that is so dependent on having a good network can be difficult to launch, because the network effects lead to a powerful barrier to entry, once it is in place, it is also hard to get rid of. Tiwana (2013b) proposes an interesting description and analysis of platforms that can be applied on blockchain technology (and blockchain-based smart contracts). If you are interested in reading more about the subject of this thesis from a platform and managerial perspective, appendix B provides a summary and analysis.

Another issue is that adopting a new technology requires some effort, regardless of how easy the new technology is in its use. The usefulness of a technology has a significant larger effect on the attitude towards a new information technology (and thus on the potential adoption of it), than the ease of use of the technology (Davis, 1989; Karahanna, Straub, & Chervany, 1999). The usefulness of the technology was a significant factor for the continued use of the technology as well. Similarly, The support of top management is necessary for successfully adopting a new technology (Karahanna et al., 1999; Liang et al., 2007).

The issue of governance remains strong due to the distributed nature of blockchain technology. Questions like 'how are decision rights distributed among the users' and 'what types of control mechanisms do the (possible) owners have' are important. In the case of a public, permissionless blockchain like Ethereum, there are no owners, and decision rights are distributed equally among all users. However, if one is for example to consider a private, permissioned, blockchain technology network, then there are one or more selected decision makers on the network that do have control mechanisms over the other users. Then, for trust in the network, the decision rights should be distributed

in a way that it remains advantageous to join the network, even if one does not have any actual power over the development and decision making in the network.

A final issue, that can be considered the most difficult as well, is that the embedded business processes need to be aligned with the new technology (Liang et al., 2007). This is also aligned with the need for top management support for the technology to be successfully adopted. However, adopting a new institutional technology (like blockchain technology) means that you change the way something is done. As was discussed, a new technology becomes an institutional technology if it provides a decrease in transaction costs. Thus, if the company adopts a new (institutional) technology, it needs to be certain it does actually provide a decrease in transaction costs. If the technology appears to not do that, then the company could have already set itself back severely with all the time spend on developing and implementing the technology. Thus the consequences are either very positive or very negative, and a lack of information beforehand does not allow to properly assess which one it is, which makes it difficult for management to take the plunge towards a new technology.

7. Summary and conclusions

The main issue when discussing and analysing blockchain technology is that it, to use the words of my thesis supervisor, is 'a technology without a problem.' This statement may seem confusing, as I have discussed several issues with blockchain technology and smart contracts. Instead, this statement refers how blockchain technology still lacks a specific application: there is not yet a specific problem that it is used for to solve. Yes, in the form of cryptocurrency it can be used as an alternative for banks, however, there are arguments against the practicality of this application. For example, the energy consumption of this system tends to be significant (Roubini, 2018), and it is slower than conventional methods. However, this is an application of blockchain technology. Because blockchain technology in and of itself does not have a specific application or purpose, this makes it difficult to analyse. Therefore, significant amounts of research are focussed on finding a problem for blockchain technology to solve. This paper is no different in that regard.

In the previous sections six sub-questions have been answered, with the purpose of becoming able to answer the main research question. It was found that there are two types of contracts on three levels. On the theoretical level there are complete and incomplete contracts, on the conceptual level there are code-only and ancillary smart contracts, and finally on the empirical level there are technical and performance contracts. On one hand, as was found in Table 5, a technical contract best fits the description of a complete contract, as does a code-only smart contract. Therefore, code-only smart contracts are the best fit for technical contracts. On the other hand, a performance contract best fits the description of an incomplete contract, as does an ancillary smart contract. Thus ancillary smart contracts are the best fit for performance contracts.

However, it was also argued that the actual use of code-only smart contracts can be difficult. This is because of the non-operational aspects of a contract that cannot be put into a smart contract, and because transaction costs hinder the actual writing of a complete contract. Even though complete contracts, like code-only smart contracts, are considered to be the best in decreasing opportunism, the obstacles to writing such a contract are significant. I do think that writing complete contracts in an electronic format will become easier than before, especially because of the standardisation and digitalisation that are necessary to occur before the actual automation in a smart contract. Through experience it will become easier for companies to include clauses that describe situations that are unlikely to occur, making the writing of a complete contract simpler and cheaper over time. Perhaps it will even be possible to describe the non-operational aspects of contracts in the digital format. Therefore, I think that ancillary smart contracts will be the norm in the beginning, while the industry is learning. Then over time, code-only contracts will become more and more common to use.

Regarding the operation and maintenance industry, five issues could be identified in the contracting process. The three communication issues are:

(1) communication between organisational layers of contracting parties in case of conflict and a need for escalation,
(2) the correct wording of contract specifications, and
(3) the need for fast data sharing in a project with many parties.

The two standardization issues are:

(4) project execution and data recording in and between systems, and
(5) data sharing between contracting partners.

The communication issues had a benefit of the use of a smart contract application, while the standardisation issues could be resolved with it, but another technology could do so as well. Overall, based on usefulness of the potential application, I estimate that the application of smart contracts could most definitely help resolve all five issues. There is, however, a need for proper support from top

management, and cooperation with large, influential players in the industry, to ensure adoption and diffusion of the technology. The application for working with smart contracts has yet to be developed for this specific purpose, but prototypes can be found for different industries (as is demonstrated in the adoption of Hyperledger Fabric in for example supply chain management). The managerial consequences of wanting to adopt the technology (and then proceeding to do so) will be difficult and network management will be key. The change in the technical area will be large, considering that smart contracts operate virtually in the opposite manner of text-based contracts (as can be seen in Table 6).

Overall, the main benefits of applying blockchain-based smart contracts for contracting would be increased automation, the removal of intermediaries, and a decrease in transaction costs. However, the removal of intermediaries may not be a desired effect from the perspective of a company such as Arcadis. This is because companies like Arcadis can also have the role of an intermediary in projects (J. Kooistra, personal communication, December 4, 2018). Thus adopting a technology like this, and advocating its adoption throughout the ecosystem, could lessen business.

To answer the main question "to what extent can blockchain-based smart contracts be used in Ethereum for contracting in the infrastructure division of the operation and maintenance industry?" I say, blockchain-based smart contracts can be used for contracting in the operation and maintenance industry for automation of operational aspects of contracts. During the contract execution period, it can improve especially in the area of communication. The successful adoption of the technology is automatically paired with the standardisation that is desired, at least between the participants of the network. The aspect where I have my doubts lies in Ethereum. Especially for contracting, privacy is incredibly important. The transparency of a public, permissionless network, is (arguably) not suitable for this. And while there are applications like Hawk that work to make transactions unreadable except for those with permission, these applications are still not fully developed or running. Also, in a network like Ethereum any smart contract runs on any node in the network, this along with the Proof of Work method of reaching consensus provides a network that scales badly. I believe that Ethereum functions better in the potential business-to-consumer market, than the business-to-business market as is the focus of this paper.

Overall, the issues that will mainly pose a problem for the adoption of smart contracts for business-to-business contracting in Ethereum will be the uncertain legal stance of the smart contracts, that they (possibly) cannot be changed, the 'inconsistent trinity', privacy, and possible weaknesses in the code. Fortunately, there are solutions to these problems. However, they require another type of blockchain network. From R. Vrees (personal communication, November 26, 2018) and J. Roling (personal communication, November 26, 2018), who work at IBM, I understood that these problems do not have to be an issue. In the case of Hyperledger Fabric, it is similar to Ethereum in that it provides a base upon which blockchain technology applications can be build. However, it is a private(/public) permissioned blockchain network (The Linux Foundation, 2018). It is specifically designed for business-to-business applications, which is what this paper is focusing on. In the case of Hyperledger Fabric, smart contracts can be changed as well. This is done by sending a transaction to the contract with the updated information/rules. Also, there are build-in functions to keep the contents of the contract and its transactions hidden for outsiders/others on the network, with the possibility of publishing certain information, and withholding other information. This way transparency can remain for the involved parties and to the outside, while privacy and sensitive information can be shielded. Overall, a blockchain technology based network like Hyperledger Fabric provides privacy protection, throughput speed, and the immutability of a blockchain network, while removing and/or decreasing the issues of using a public, permissionless blockchain based network like Ethereum for contracting. Considering all of the above, I think that blockchain-based smart contracts for contracting would work better in a private, permissioned blockchain network, even if this is not desired when considering the current overall need for increased transparency.

The main issue that will remain, that will be an issue regardless of whether the adopted application is based on blockchain or not, is the link between the online and offline version of an entity. While a private, permissioned blockchain network ensures that you know your transaction partner (as is the case with traditional contracts), it is still reliant on the documentation of the contract partners for its information. To prevent opportunism, validations and checks will remain necessary in (most likely) a similar form as to how they are done currently with for example, sample tests for an ancillary contract. In code-only contracts it will be necessary to build in function that allow for repercussions and prevent abuse of information asymmetry. Rules and regulations can possibly be standardised across the network, automatically adding certain regulations and methods for dealing with following issues to any smart contract created. However, it can take some time, and significant trial and error, until such regulations function as is desired.

There is also the matter of energy consumption (Roubini, 2018). This is more evident for public, permissionless blockchains due to their need for a Proof of Work method for consensus (as there is still no proper alternative found, that does not sacrifice security like Proof of Stake). Private, permissioned blockchains can adopt a different consensus mechanism that requires less energy, because it does not need to compensate for the anonymity and lack of trust between its users. However, especially where mutual trust (to some degree) between parties is important (like with a contracting) it is undesirable to implement a consensus and verification mechanism that heavily centralises the power. While it should be clear for a firm what its role is in the network, it will not want to rely too heavily on a select few large parties for deciding what happens in the network. Table 7 provides a summary of the different issues discussed and compares them between two very different blockchain technology types: public, permissionless blockchains, and private, permissionless blockchains.

Table 7 – Summary of any issues discussed above, in the format of a comparison between two blockchain technology types regarding

	Public, permissionless blockchain technology network	Private, permissioned blockchain technology network
Uncertain legal stance of the smart contracts	Yes	Yes
Smart contracts cannot be changed	Yes	Can possibly be changed, if build-in mechanism for it
The 'inconsistent trinity'	Yes	No
Privacy	No	Yes
Possible weaknesses in the code?	Yes	Yes
Governance	No, always democratic	Yes, can have different types, need to find out what type is desired
Energy consumption	Yes, due to Proof of Work method	No, due to non-democratic, centralised methods of consensus

I would like to finish this part by describing four future scenarios which I envision. The first scenario is what I deem to be the most positive scenario. In this scenario, smart contracts for contracting have spread past the operation and maintenance industry. This is mainly due to the increased willingness to provide transparency in processes by businesses, while some sensitive information remains private and shielded (like money flows). The blockchain network upon which the smart contracts operate is then (partly) public and permissioned, with the data largely being transparent, but you still need qualifications to join the network, to prevent it from growing needlessly large and maintain efficiency.

Both companies and the public sector operate on the blockchain network together, possibly individuals as well.

The second scenario is more like what I have described as a reasonable scenario above, considering the need for privacy of data for companies. In this scenario, the blockchain network remains private and permissioned. Only firms make use of it and the public sector and individuals will refrain from using it, because they prefer more transparency. The use of smart contracts for contracting might spread to other business-to-business industries, but not further.

The third scenario is the failure of the technology. Even in the case of a private, permissioned blockchain network, the firms operating on it cannot get to an agreement concerning the governance of the network. The lack of transparency bars other parties form entering and finally the project dies as conflicts remain unsolved.

A fourth scenario, which I think is most unlikely, is that the network becomes public and permissionless. Transparency will be complete, thus no data is hidden from others. Firms, the public sector, and individuals all operate on the network. And the technology becomes the new way of writing the operational aspect of contracts.

7.1 What is interesting for different industries?

Findings in this paper can still be useful for different industries. To start with, I provided a definition for blockchain-based smart contracts, and defined the assumptions behind and operational aspects of blockchain-based smart contracts. These can still be used by companies in different industries to compare their contract types to, similar to the comparison done in Table 5. The findings form Table 6 also remain valid if an industry uses traditional text-based contracts for contracting. And similarly, if it uses a different way of writing contracts, the operational aspects of smart contracts can then still be compared, just like what was done in Table 6 with text-based contracts. The issues addressed above, in Table 7, also apply to different industries, not merely the infrastructure division of the operation and maintenance industry.

Overall, the manner of analysis used in this paper can be copied fairly easily, provided that the company operates in a business-to-business environment. The main focus is then for the company to identify and describe the contracts it uses, and express them in the terms of contracting theory regarding the assumptions behind the contracts, and in the operational terms used to describe smart contracts.

Regarding the actual application of blockchain-based smart contracts, it is especially important, regardless of the transaction environment (business-to-business, business-to-consumer, etc.), that partners to establish the network with are found first. Specifically focus on collaborating with large players in the industry, because this enhances the diffusion of the technology. In appendix B the platform perspective is highlighted and applied, which provides some more insight and information regarding the potential adoption and evolution of the network.

7.2 Limitations

In this paper, I argued that I consider blockchain technology an institutional technology. This indirectly also means that it can be used for more innovative purposes than just more efficient production processes. However, from a company perspective, it is often unclear what blockchain technology can do, or what its possibilities are. This translates to the phenomenon that while I analyse it like an institutional technology, companies often treat it like a general-purpose technology (De redactie, 2018). While understandable, this view indirectly restricts the range of innovations that are considered and looked at with the technology to areas where it can improve productivity. The smart contracts application I observed and analysed can be looked at from both perspectives: from the institutional perspective it provides a new way of writing and executing contracts. From a general-purpose technology perspective, it is a way of increasing the efficiency of creating and executing contracts.

While this latter perspective is not wrong (it is, after all, the function of the application), I still feel it falls short on what the application encompasses. Of course, in order to be adopted it needs to do something better than the old institution, but only defining it by this is, in my opinion, inadequate.

Also, I almost exclusively spoke with people employed by Arcadis Nederland B.V. This limited the information I got to the perspective of a single (albeit large) company present in the sector. More research is necessary to get a more accurate depiction of the industry and its needs and wants, along with the willingness to adopt and possible diffusion rate of the platform. This includes for example government agencies, but also small and larger service providers. Each side can have its own needs, wants, and obstacles for adopting the technology, that can differ significantly from each other.

What is also a limitation is that due to the use of Transaction Cost Economics I only focussed on the application of blockchain-based smart contracts for contracting in the business-to-business environment. Therefore, my analysis will not apply to other environments, such as business-to-consumer. Some estimations can be made for other environments, for example, while Ethereum does not have a preference in a business-to-business environment, it could work perfectly fine for for example business-to-consumer. Further research will be required to answer (an adaptation of) my research question for these environments.

This also leads to a limitation in terms of how blockchain technology and smart contracts actually deal with opportunism. I have made some arguments as to how smart contracts deal with it, but not for blockchain technology in and of itself. This area is still largely unexplored and further research will be necessary how, why, and if blockchain technology and smart contracts actually decrease opportunism.

Another limitation is that I only focused on the two extremes of the contract types in a specific industry division: performance vs technical. There is a range of contracts in between the two extremes, and in the case of other industries, contracts may be too different from the contracts I observed for this analysis to apply to. However, I think that the preference of ancillary smart contracts over code-only smart contracts will remain, regardless of the industry and as long as at least one non-operational aspect needs to be specified.

A final limitation is concerning the government. The public sector desires more transparency than the private sector, because it works with public money. This means that the government needs to be able to show the people what their money is being used for, and how. Money flows are a sensitive subject for companies, which is why one of the possible future scenarios that I describe keeps the network private and permissioned, and only used by companies, or another where certain information is still private. However, especially in the case of the operation and maintenance industry, where the government is a large player in for example infrastructure jobs, it is desired that it participates in the network. Then, the firms will need to accept more transparency by themselves and others. The question is: will they be able to get outside of their current, private, comfort zone, and take the leap? More research will be necessary to answer this question, along with time to observe the possible shift in transparency needs by the people.

Overall, many questions remain that I cannot answer or research. This research was done in a new area, with a focus different from conventional research regarding blockchain technology and smart contracts. Further research will be necessary to properly address questions regarding different industries and applications. There is also a need for more research on the definition of smart contracts, the different types, and how opportunism can be decreased using them. I hope that I have shed some light on an area still mostly unexplored, and provided a first step in a new research direction.

Bibliography

Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., … Zanella-Beguelin, S. (2016). Formal Verification of Smart Contracts. *Proceedings of the 2016 Acm Workshop on Programming Languages and Analysis for Security (Plas'16)*, 91–96. https://doi.org/10.1145/2993600.2993611

Bresnahan, T. F., & Trajtenberg, M. (1995). General purpose technologies "engines of growth"? *Journal of Econometrics*, *65*(1), 83–108.

Buterin, V. (2013). *Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform*. *Ethereum wiki*.

Buterin, V. (2016). Privacy on the Blockchain. Retrieved November 5, 2018, from https://blog.ethereum.org/2016/01/15/privacy-on-the-blockchain/

Buterin, V. (2018). twitter.com. Retrieved October 23, 2018, from https://twitter.com/VitalikButerin/status/1051160932699770882

Catalini, C., & Gans, J. S. (2017). *Some Simple Economics of the Blockchain* (No. 2874598). https://doi.org/10.2139/ssrn.2874598

Christidis, K., & Devetsikiotis, M. (2016). Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, *4*, 2292–2303. https://doi.org/10.1109/ACCESS.2016.2566339

Clack, C. D., Bakshi, V. A., & Braine, L. (2016). *Smart Contract Templates : foundations , design landscape and research directions*. Retrieved from https://arxiv.org/abs/1608.00771

Coase, R. H. (1937). The Nature of the Firm. *Economica*, *4*(16), 386–405.

Dai, P., Mahi, N., Earls, J., & Norta, A. (2017). *Smart-Contract Value-Transfer Protocols on a Distributed Mobile Application Platform*. https://doi.org/10.13140/RG.2.2.35140.63365

Davidson, S., De Filippi, P., & Potts, J. (2018). Blockchains and the economic institutions of capitalism. *Journal of Institutional Economics*, *14*(4), 639–658. https://doi.org/10.1017/S1744137417000200

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, *13*(3), 319–340. https://doi.org/10.2307/249008

De redactie. (2018). Impact blockchain in Europa belemmerd door gebrek aan samenwerking. Retrieved October 30, 2018, from https://accountantweek.nl/artikel/impact-blockchain-in-europa-belemmerd-door-gebrek-aan-samenwerking-

Dedaub. (n.d.). Contract Library. Retrieved January 4, 2019, from https://contract-library.com/#/

Duchenne, J. (2018). *Blockchain and Smart Contracts. Transforming Climate Finance and Green Investment with Blockchains*. Elsevier Inc. https://doi.org/10.1016/B978-0-12-814447-3.00022-7

Frantz, C. K., & Nowostawski, M. (2016). *From Institutions to Code: Towards Automated Generation of Smart Contracts*.

Governatori, G., Idelberger, F., Milosevic, Z., Riveret, R., Sartor, G., & Xu, X. (2018). On legal contracts, imperative and declarative smart contracts, and blockchain systems. *Artificial Intelligence and Law*, *26*(4), 377–409. https://doi.org/10.1007/s10506-018-9223-3

Hart, O. D. (1989). An Economist's Perspective on the Theory of the Firm. *Columbia Law Review*, *89*(7), 1757–1774.

Hart, O. D., & Moore, J. (1990). Property Rights and the Nature of the Firm. *Journal of Political Economy*, *98*(6), 1119–1158.

Hart, O. D., & Moore, J. (1999). Foundations of Incomplete Contracts. *The Review of Economic Studies*, *66*(1), 115–138.

Hofs, Y., & Misérus, M. (2018). Kustwacht koopt twee keer kat in de zak met dure speedboten. *De Volkskrant*. Retrieved from https://www.volkskrant.nl/nieuws-achtergrond/kustwacht-koopt-twee-keer-kat-in-de-zak-met-dure-speedboten~b6e62220/

Jayachandran, P. (2017). The difference between public and private blockchain. Retrieved January 4, 2019, from https://www.ibm.com/blogs/blockchain/2017/05/the-difference-between-public-and-private-blockchain/

Kandampully, J., & Duddy, R. (1999). Competitive advantage through anticipation, innovation and relationships. *Management Decisions*, *37*(1), 51–56. https://doi.org/10.1108/00251749910252021

Karahanna, E., Straub, D. W., & Chervany, N. L. (1999). Information Technology Adoption Across Time : A Cross-Sectional Comparison of Pre- Adoption and Post-Adoption Beliefs. *MIS Quarterly*, *23*(2), 183–213. https://doi.org/10.2307/249751

Kosba, A., Miller, A., Shi, E., Wen, Z., & Papamanthou, C. (2016). Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016* (pp. 839–858). https://doi.org/10.1109/SP.2016.55

Kshetri, N. (2017). Blockchain's roles in strengthening cybersecurity and protecting privacy. *Telecommunications Policy*, *41*(10), 1027–1038. https://doi.org/10.1016/j.telpol.2017.09.003

Kwak, K. H., Kong, J. T., Cho, S. I., Phuong, H. T., & Gim, G. Y. (2019). A study on the design of efficient private blockchain. *Studies in Computational Intelligence*, 93–121. https://doi.org/https://doi.org/10.2139/ssrn.2874598

Laffont, J.-J., & Martimort, D. (2001). *THE THEORY OF INCENTIVES I : THE PRINCIPAL-AGENT MODEL*.

Lauslahti, K., Mattila, J., & Seppälä, T. (2017). *Smart Contracts – How will Blockchain Technology Affect Contractual Practices ?* (No. 68). Retrieved from https://www.etla.fi/wp-content/uploads/ETLA-Raportit-Reports-68.pdf

Levi, S. D., & Lipton, A. B. (2018). An Introduction to Smart Contracts and Their Potential and Inherent Limitations. Retrieved November 5, 2018, from https://www.lexology.com/library/detail.aspx?g=683fb3c1-dd6f-4daa-9fc7-dd43d44dcd24

Li, W., Sforzin, A., Fedorov, S., & Karame, G. O. (2017). Towards Scalable and Private Industrial Blockchains. *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, 9–14. https://doi.org/10.1145/3055518.3055531

Liang, Saraf, Hu, & Xue. (2007). Assimilation of Enterprise Systems: The Effect of Institutional Pressures and the Mediating Role of Top Management. *MIS Quarterly*, *31*(1), 59–87. https://doi.org/10.2307/25148781

Lind, H., & Nyström, J. (2011). The Explanation of Incomplete Contracts in Mainstream Contract Theory: A Critique of the Distinction between "Observable" and "Verifiable." *Evolutionary and Institutional Economics Review*, *7*(2), 279–293. https://doi.org/10.14441/eier.7.279

Luu, L., Chu, D.-H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making Smart Contracts Smarter. In *Conference on Computer and Communications Security* (pp. 254–269). Vienna: ACM. https://doi.org/10.1145/2976749.2978309

Ménard, C., & Shirley, M. (2005). Transaction Cost Economics. In *Handbook of New Institutional Economics* (pp. 41–65). https://doi.org/10.1080/03585520903122574

Nelson, R. R., & Sampat, B. N. (2001). Making sense of institutions as a factor shaping economic

performance. *Journal of Economic Behavior and Organization*, *44*(1), 31–54. https://doi.org/10.1016/S0167-2681(00)00152-9

Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, *59*(3), 183–187. https://doi.org/10.1007/s12599-017-0467-3

Olleros, F. X., & Zhegu, M. (2016). *Research Handbook on Digital Transformations - Google Boeken*. Edward Elgar Publishing. https://doi.org/10.4337/9781784717766

Parmar, R., Mackenzie, I., Cohn, D., & Gann, D. (2014). The new patterns of innovation. *Harvard Business Review*, (JAN-FEB).

Puthal, D., Malik, N., Mohanty, S. P., Kougianos, E., & Yang, C. (2018). The Blockchain as a Decentralized Security Framework [Future Directions]. *IEEE Consumer Electronics Magazine*, *7*(2), 18–21.

Reijers, W., O'Brolcháin, F., & Haynes, P. (2016). Governance in Blockchain Technologies & Social Contract Theories. *Ledger*, *1*, 134–151. https://doi.org/10.5195/LEDGER.2016.62

Ritter, T., & Gemünden, H. G. (2003). Network competence : Its impact on innovation success and its antecedents. *Journal of Business Research*, *56*, 745–755. https://doi.org/10.1016/S0148-2963(01)00259-4

Roubini, N. (2018). *Testimony for the Hearing of the US Senate Committee on Banking, Housing and Community Affairs On "Exploring the Cryptocurrency and Blockchain Ecosystem."*

Savelyev, A. (2017). Contract law 2.0: 'Smart' contracts as the beginning of the end of classic contract law. *Information and Communications Technology Law*, *26*(2), 116–134. https://doi.org/10.1080/13600834.2017.1301036

Schmitz, P. W. (2001). *The Hold-Up Problem and Incomplete Contracts: A Survey of Recent Topics in Contract Theory* (No. 12562).

Schwartz, A., & Scott, R. E. (2003). Contract Theory and the Limits of Contract Law. *Faculty Scholarship Series*, 541–619. Retrieved from http://digitalcommons.law.yale.edu/fss_papers/308

Scott, R. E. (2003). A THEORY OF SELF-ENFORCING INDEFINITE AGREEMENTS. *Columbia Law Review*, *103*(7), 1641–1699.

Scott, R. E., & Triantis, G. G. (2005). Incomplete Contracts and the theory of Contract Design. *Case Western Reserve Law Review*, *56*(1), 187–201. https://doi.org/10.3366/ajicl.2011.0005

Searls, D. (2003). Closing the Chasm. Retrieved December 5, 2018, from https://www.linuxjournal.com/article/6629

Smith, Y. (2003). Understanding reliability and validity in qualitative research. *The Qualitative Report*, *8*(4), 597–607. https://doi.org/10.3367/UFNr.0180.201012c.1305

Szabo, N. (1994). Smart Contracts. Retrieved October 18, 2018, from http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

Tapscott, D., & Tapscott, A. (2016). Blockchain Revolution. In *Blockchain Revolution* (pp. 3–52). Retrieved from http://www.newparadigm.com/media/tapscott_bio.pdf

The Linux Foundation. (2018). Hyperledger Fabric – Hyperledger. Retrieved November 27, 2018, from https://www.hyperledger.org/projects/fabric

Tirole, J. (1999). Incomplete Contracts: Where Do We Stand? *Econometrica*, *67*(4), 741–781.

Tiwana, A. (2013a). Chapter 1: The Rise of Platform Ecosystems. In *Platform Ecosystems* (pp. 3–21). https://doi.org/10.1016/B978-0-12-408066-9.00001-1

Tiwana, A. (2013b). Chapter 2: Core Concepts and Principles. In *Platform Ecosystems: Aligning Architecture, Governance, and Strategy* (pp. 23–48). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-408066-9.00002-3

Tiwana, A. (2013c). Chapter 6: Platform Governance. In *Platform Ecosystems* (pp. 117–151). https://doi.org/10.1016/B978-0-12-408066-9.00006-0

Tiwana, A., Konsynski, B., & Bush, A. A. (2010). Platform Evolution : Coevolution of Platform. *Information Systems Research*, *21*(4), 675–687. https://doi.org/10.1287/isre.1100.0323

Tjong Tjin Tai, E. (2017). Smart contracts en het recht. *Njb*, *92*(3), 176–183. Retrieved from http://www.narcis.nl/publication/RecordID/oai%3Atilburguniversity.edu%3Apublications%2Fa4f7d1c0-ee74-417e-8508-d721db602ff0

Vaibhav Saini. (2018). Getting Deep Into Ethereum: How Data Is Stored In Ethereum? Retrieved February 20, 2019, from https://hackernoon.com/getting-deep-into-ethereum-how-data-is-stored-in-ethereum-e3f669d96033

Vraalsen, F., Aydt, R. A., Mendes, C. L., & Reed, D. A. (2007). Performance Contracts: Predicting and Monitoring Grid Application Behavior, 154–165. https://doi.org/10.1007/3-540-45644-9_15

Williamson, O. E. (1979). Transaction-Cost Economics: The Governance of Contractual Relations. *Journal of Law and Economics*, *22*(2), 233–261.

Williamson, O. E. (1991). Comparative Economic Organization : The Analysis of Discrete Structural Alternatives. *Administrative Science Quarterly*, *36*(2), 269–296.

Williamson, O. E. (2008). Outsourcing: Transaction Cost Economics and Supply Chain Management. *Journal of Supply Chain Management: A Global Review of Purchasing & Supply*, *44*(2), 5–16. https://doi.org/doi:10.1111/j.1745-493X.2008.00051.x

Wong, I. W., & Kar, I. (2016). The Ethereum hard fork: Everything you need to know. Retrieved November 22, 2018, from https://qz.com/730004/everything-you-need-to-know-about-the-ethereum-hard-fork/

Xu, X., Pautasso, C., Gramoli, V., Ponomarev, A., Binh Tran, A., & Chen, S. (2016). The Blockchain as a Software Connector. In *IFIP Conference on Software Architecture (WICSA)* (pp. 182–191). Retrieved from https://www.ascribe.io/

Yin, R. K. (1992). The Case Study Method as a Tool for Doing Evaluation. *Current Sociology*, *40*(1), 121–137. Retrieved from https://doi.org/10.1177/001139292040001009

Young, J. C., Rose, D. C., Mumby, H. S., Benitez-Capistros, F., Derrick, C. J., Finch, T., … Mukherjee, N. (2018). A methodological guide to using and reporting on interviews in conservation science research. *Methods in Ecology and Evolution*, *9*(1), 10–19. https://doi.org/10.1111/2041-210X.12828

Appendix A – Interview guide

The interview guide in Table 8 used contained a number of focus areas, with questions that were asked per area. Because interviewees could have specific knowledge in some areas, but no clue in other areas, which focus areas were explored in an interview were dependent on the interviewee. Generally, one question was asked per focus area, and when the interviewee showed he/she had no knowledge regarding this area, the next one was explored.

Table 8 – Interview guide

Focus area	Questions
The operation and maintenance industry	How do you define the operation and maintenance industry?
	Are you familiar with opportunism?
	What types of contracts are commonly used by Arcadis in this industry?
	How do you define a performance contract?
	How do you define a technical contract?
Contracting	Can you explain how the contracting process currently goes?
	What kind of aspects in a contract would purposefully be kept vague or undefined?
	What information is considered too sensitive to be observable by external parties, not involved in the contract?
	With differences are there in terms of desired information between the contracting parties, if there is any?
	What parts of the process would benefit from increased transparency?
	How often is renegotiation required? Are the number of renegotiations decided on at the time of contracting?
	What processes in the contract are automated, if there are any?
Opportunism	In what areas do you consider opportunism to be an issue in the contracting process?
	What are common ways in which parties try to take advantage of a situation?
	What processes or measures are currently used in order to minimise opportunism?
	What happens in case of escalation?
Blockchain technology	What do you know of blockchain technology?
	What is your impression of blockchain technology?
	What do you think are benefits of blockchain technology?
	What do you think are disadvantages of blockchain technology?
	If you were to apply blockchain technology, what kind of applications come to your mind?
	Do you know of any projects concerning blockchain technology within either Arcadis or the industry?

Smart contracts	What do you think of when you hear ‘smart contract’?
	How do you think that an application using smart contracts on blockchain technology will be received by contracting partners (assuming that you have a finished application and offer its use to them)?
	How would you feel about smart contracts that are immutable? Any advantages or disadvantages that come to mind?
Standardisation and digitalisation	Are standardisation and digitalisation high in priority for current developments within the company and industry?
	What kind of programs or technologies are currently used for process recording and communication?
	What are the main obstacles for development for new programs or technologies to increase standardisation and digitalisation?

Appendix B – Platform perspective

When a company decides to adopt blockchain-based smart contracts for contracting, it is effectively adopting a new software platform for contracts. This is because it is an institutional technology, and thus provides a new way of doing something, in this case contracting. And when it comes to adopting new platforms, there are some aspects that managers need to be aware of, before deciding on whether to make the decision.

To start with, what is a platform? I adopt the vision of a platform from Tiwana (2013a, 2013b). Here, a (software) platform is described as something that brings two distinct groups together. For example, the android platform brings together app developers on one side, and the end users on the other side. If I were to apply this to the adoption of blockchain-based smart contracts in the operation and maintenance industry, then the two sides that are brought together are the clients, and the contractors. However, some parties can be both, depending on the contract (Arcadis is an example of this). Thus, the distinction between the two groups can change with each contract. Also, a company can have multiple contracts running simultaneously, and thus also operate on both sides of the platform simultaneously. I still want to argue that the vision of Tiwana (2013a, 2013b) can be used for insight on platform architectures, governance, and evolution. Because describing these factors does not necessarily depend on the presence of two distinct groups on the sides of the platform, of which the users cannot change sides or occupy roles on both sides.

A platform ecosystem consists of two elements: a platform, and complementary apps (Tiwana, 2013a). The platform is the blockchain network, the complementary app I specifically look for is an application that creates smart contracts on the platform, which can be read and adapted with the application as an interface. This means that multiple versions of an interface can be created that create smart contracts on the platform, but that can also do other things. However, an application can also become a platform, in the sense that a dominant design can be established regarding what the application does. Therefore, the next parts on how to identify the platform's lifecycle, how they begin and evolve, and guiding principles for platforms, can be applied to both the blockchain network, and the application that works as an interface to the smart contracts on the blockchain network.

Platform lifecycle

Then, to describe a platform, it is important to first start with the platform lifecycle: where is it? A platform's stage can be described using three dimensions, depicted in Figure 7.

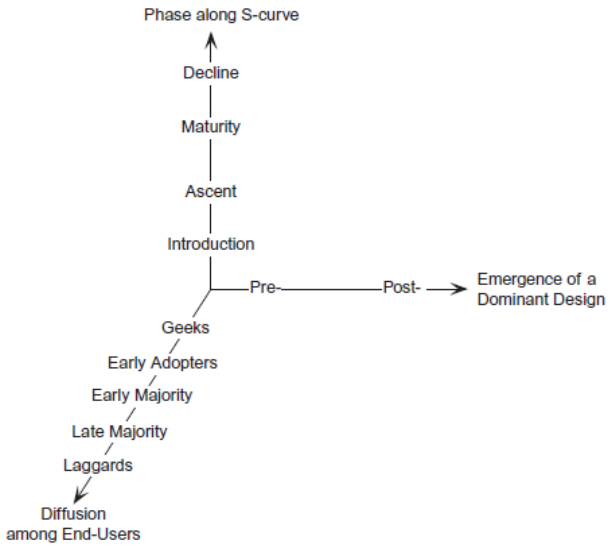


Figure 7 - The three dimensions of a platform lifecycle (Tiwana, 2013b)

Figure 8 shows a depiction of the two design phases of a software platform. During the pre-dominant design phase, there are many competitors working with many alternative designs. At some point, a dominant design emerges, and the number of competitors decreases. While there can still be alternative designs in the post-dominant design phase, however, these often share strong commonalities with the dominant design.

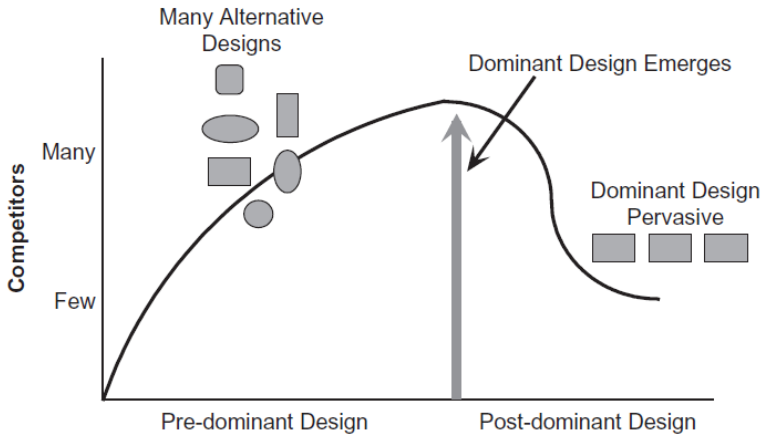


Figure 8 - Pre- and Post-dominant design phases in a software platform (Tiwana, 2013b)

A dominant design emerges because of two primary reasons (Tiwana, 2013b). First, there is often what you call increasing returns to adoption. This means that the more a technology is adopted, the more valuable it becomes. And also, the more a design is used, the more it is improved. The second reason is that as a platform is adopted more widely, there come more specialised compliments that are designed to work specifically for that platform as well. This makes the platform more attractive to use and results in a positive feedback loop which makes the platform increasingly more dominant. It is of no relevance whether competing designs are technologically superior once this process is in motion.

The second dimension to observe is the platform's phase on the s-curve. Figure 9 depicts this curve, along with how one can 'leapfrog' to the S-curve of the next technology solution. The s-curve itself exists out of these phases, in chronological order: introduction, ascent, maturity, and decline. Thus the goal is to leapfrog from one s-curve to the next when the first s-curve is starting towards the decline phase, and the next one is starting to reach maturity. However, most companies fail doing this leap, because they are too invested in the current (declining) technology.

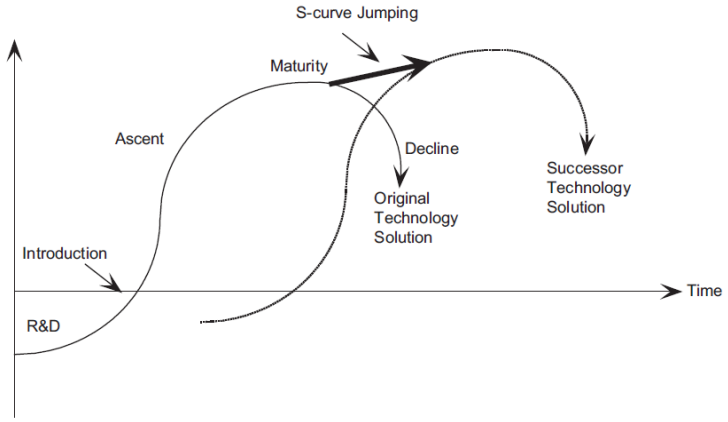


Figure 9 - S-curves in the technology lifecycle (Tiwana, 2013b)

To describe the adoption of smart contracts in both ways to define its lifecycle phases, it is in the pre-dominant design phase. This is because there are many blockchain networks present, and while you may say that all networks are quite similar, there is still a grave discussion and many weaknesses in each platform version. For example, a public, permissionless network such as Ethereum suffers from scalability issues and an enormous energy consumption. While a private, permissioned blockchain such as Hyperledger suffers from a lack of research into its boundaries and effectiveness. Also, there is still the general debate regarding what the best way is to create consensus on a blockchain network, since both Proof of Work and Proof of Stake have their own weaknesses. In the case of blockchain-based smart contracts for contracting, there is barely any evidence of actual application, therefore, it is still in the pre-dominant design phase as well. However, it is in an earlier stage of this phase than blockchain technology.

While many people have heard of blockchain technology, and then especially through bitcoin, there are not many people who use it. Especially established companies are only just starting to launch applications of the technology, often on private, permissioned blockchain platforms. Personally, I think that blockchain technology is still in its introduction phase based on the second way. Especially considering the limited real-world applications that are successful, and the vast potential it still holds. Smart contracts are in a similar place, although they do see some use, it is not in the manner this paper explores. I want to say it is in its introductory phase on the s-curve, considering the emergence of networks like Hyperledger. However, it can also be placed in the R&D (Research & Development) phase, considering the lack of actual application, and the heavy research and development in this area (the use of smart contract by companies for contracting).

The third and final dimension to analyse is the diffusion among end users, illustrated in Figure 10. Although large companies such as IBM and banks are starting to invest in blockchain and use it, the successful adoption rate by companies is still low. Overall, it still lacks any mainstream recognizability. Therefore, I categorize the platform in the area of early adopters, which means that the success of the platform depends on whether it can successfully cross the chasm towards early majority. Where innovators/geeks and early adopters desire to be first users of a breakthrough technology and will pay for such a thing (even if it still has some performance issues), the early majority is more risk averse and wants a cheaper, properly functioning technology before they adopt it. This difference in expectations of the two groups makes for an incompatibility in terms of marketing, performance, and price when selling the product. Smart contracts in the manner of which this paper wants to use them do not have a working version at this moment. It is still mostly people figuring out whether they can be used for this, therefore, any adopters of an application that works with smart contracts in this manner is for the innovators/geeks. Similarly, I think blockchain technology is still for the early adopters, and has yet to successfully cross the chasm. However, I do also think that it is close to doing so in certain industries (e.g. fairtrade and cryptocurrency).

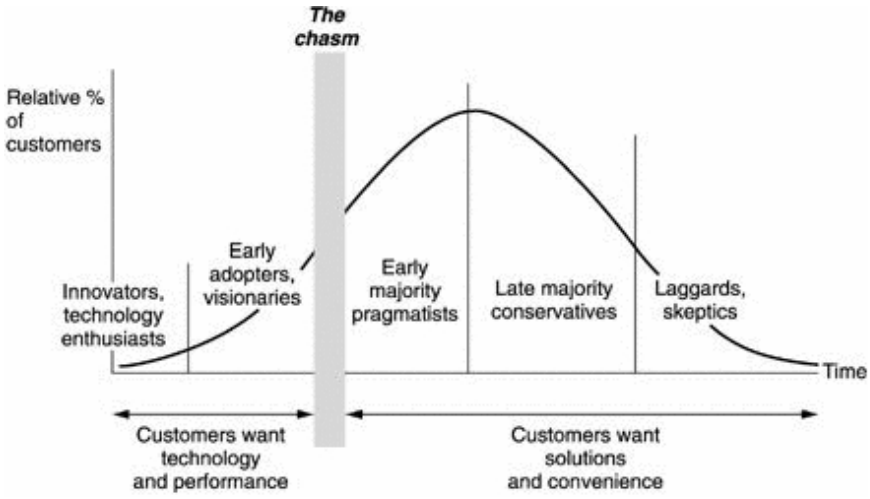


Figure 10 - Diffusion of innovations (Searls, 2003)

To conclude, blockchain technology is a platform still in a pre-dominant design and introduction on the s-curve phases, and the diffusion of the innovation is still among the early adopters. The application for the blockchain-based smart contracts is also still in its pre-dominant design phase and introduction on the s-curve phases (maybe even still in R&D), the diffusion of the technology is still among the innovators (or geeks). Thus it is slightly behind the blockchain network in its lifecycle. Overall, the application of blockchain-based smart contracts is in a very early phase of its lifecycle, on both areas of technology that is necessary to make it work.

How do software platform ecosystems begin and evolve?

Next it is necessary for a manager to consider how platform ecosystems begin and evolve. Especially if you are part of the initiative to make it happen, these aspects influence strategy.

A first key notion in understanding platforms is that it creates value by facilitating participants on one side finding those on the other side (Tiwana, 2013b). The platform is valuable to both sides of the platform, by making it easier and cheaper to interact in both directions. A platform is multisided if "*two or more distinct types of participant groups use the platform to directly interact with each other*" (Tiwana, 2013b, p. 32). In the case of blockchain-based smart contracts, the clients and contractors do interact directly with each other via the platform (specifically via the app interface, interaction on the blockchain network is due to the functioning of the app on this network). However, there is also likely to be interaction outside of the platform (due to the most likely adoption of ancillary smart contracts). Thus for the application and the blockchain network to become new platform(s) for contracting, they need to facilitate communication between clients and contractors.

Another notion from a platform is network effects (or network externalities, as economists call these) (Tiwana, 2013b). This effect refers to "*the degree to which every additional user of a platform or app makes it more valuable to every other existing user*." (Tiwana, 2013b, p. 33) The mechanic is simple: every additional user to the platform represents another user that current users can interact with. And on the other hand, the more users are on the network, the more benefit they provide to anyone joining. Thus, in the operation and maintenance industry, every additional user of the application with which you create and maintain contracts, represents another user for which the benefits of using the platform count. And for communication and speed, it is useful when each contracting partner uses the same platform in a project. Therefore, the more users of the platform, the more projects can be done with it and the more users will join. While a platform can be difficult to launch, because the network effects lead to a powerful barrier to entry, once the platform is in place, it is also hard to get rid of.

Also, the adoption of a new platform does not mean that other platforms have to be ignored. Multihoming refers to when a platform user participates in multiple platforms (Tiwana, 2013b). Especially when there is not yet a clear dominant platform (form), multihoming will occur. The challenge here lies in obtaining enough users of the platform to reach a certain tipping point, from which point on the network effects become noticeable. Once a platform has reached its tipping point, harnessing its users to nudge the platform's evolutionary technology is the key challenge. Therefore, the required strategies are vastly different depending on whether the platform is before or after its tipping point. In the case of adopting blockchain-based smart contracts for contracting, Arcadis would be the first user, together with any partners with which Arcadis works together to develop this application. Then, the strategy should be first to recruiting new users for the platform, attempting to get it to its tipping point. Then, once this tipping point is reached, focus can shift more towards improvement of the platform and its application.

In the case of a software platform as I am discussing, it is desirable to also create a value-driven lock-in mechanism (Tiwana, 2013b). This refers to the phenomenon that the platform becomes more and more valuable to its users, making switching to another platform undesirable. Depending on the phase of the lifecycle in which the platform is, different strategies to achieve value-driven lock-in are required by the platform owners to influence the app developers and end users. This lock-in mechanism is desired, because once a platform starts to take off, it means that it does something more effectively than other platforms, or found a new market to serve. Competition will then increase as copy-cats start to enter the market, as well as alternatives.

A platform should also be competitively durable (Tiwana, 2013b). This means that the platform will be used for a long time after its initial adoption, with new users still adopting it, and old users not abandoning it. If Arcadis is to spearhead an institutional innovation such as a new platform for contracting, based on blockchain technology, then it should thus also think about what it can offer, or will need to keep doing together with other users, in order to remain competitively durable.

A platform can also envelop the functionality of another platform (Tiwana, 2013b). For example, in the beginning there is the blockchain-based smart contracting platform which only works to execute certain automated parts of the contract. Then, it also becomes capable of automatically printing progression reports, which was always done using another platform. Then, the blockchain platform has swallowed (enveloped) the other platform (which may still exist in parallel).

It is also important that one understands the architecture of the platform (Tiwana, 2013b; Tiwana, Konsynski, & Bush, 2010). The architecture is a high-level description of the platform's building blocks and how they are related to each other. The architecture both defines the platform itself, and define individual apps and how they interact with other apps and the platform itself. The architecture consists of mostly technical decisions, but it has significant strategic impact. For example, when smartphones became capable of making photos, they enveloped the digital camera market. Architectural decisions are also mostly irreversible in practice, meaning that any constraints they impose that become visible long after the choice was made, cannot be fixed. The challenge here is that the architectural choices need to accommodate future changes that cannot be foreseen at the time that the platform is created, constricted by the (almost definitive) irreversibility of these choices (Tiwana et al., 2010).

Then finally, there is the issue of governance (Tiwana, 2013b, 2013c; Tiwana et al., 2010), or generally stated: who decides what in a platform's ecosystem. There are three aspects to governance:

(1) How are decision rights divided between the platform owner and app developers?
(2) What types of formal and informal control mechanisms are used by the platform owner?
(3) Pricing structures, which includes decisions on whether a side should be subsidised and if so, which side?

When applying these aspects to blockchain-based smart contracts, it becomes apparent that this is not a traditional platform. For starters, decision rights are dependent on the underlying blockchain platform, and there is never one specific 'platform owner'. In the case of permissioned blockchain networks, it is possible that only a few users have decision rights in the platform, however, the idea behind blockchain is that there is no owner. Therefore, I would like to say that in this case, the platform owner can be considered the ones with decision rights. This can be everybody in the case of a public permissionless blockchain network, but also only a select few when looking at a private permissioned platform (even though this does not need to be the case per se). Control structures are not interesting, because there is no platform owner external to the platform users on both sides, who exercises control on both the users and the platform. Similarly, this is also why pricing structures are not interesting to look at in this application. Governance is a difficult question in the case of blockchain-based smart contracts, and the framework by Tiwana is not suitable to provide an answer in this area.

Principle guiding initial development and further evolution of the software platform

Now the principles on how to describe a platform are clear, there is the question on how to manage a platform. At first, one has to find a strategy for the initial development of a platform. Then, there is also a need for strategies to evolve the platform. This second part is slightly more tricky due to the underlying blockchain technology. As was discussed earlier in this paper, the blockchain platform itself cannot be changed unless an hard fork is made, after which both the new and old version exist simultaneously. However, applications and smart contracts on the platform, can be changed (if it is allowed in the platform rules). Also, it is possible to build multiple applications, so even if changing is not allowed, one can delete it (if such a thing is built into the application) and activate a new application with improved functioning. The data is still available on the blockchain network, so nothing is lost as long as the new application is compatible with, and has access to, the data.

There are four principles related to the initial development of platform ecosystems (Tiwana, 2013c, 2013b):

(1) **The red Queen Effect**: the platform system needs to evolve at least as fast as its competitors in order to survive;
(2) **The chick-or-egg problem**: the platform ecosystem needs to simultaneously attract users for both sides, while one will only join when there are enough users on the other side
(3) **The penguin problem**: when potential adopters (with potentially strong network effects) stall adopting the platform because they are unsure of whether others will adopt it too;
(4) **Emergence**: most of the innovations on the platform emerge spontaneously due to the pursuit of individual interests by users, based on their own expertise, but adapted to what other participants are doing in the platform ecosystem.

All four principles above are relevant for the blockchain-based smart contracts to take off, since the platform ecosystem consists of both the blockchain network and the application that works as an interface between the users and the smart contracts on the network. It is up to the platform owner to try and attract new users and develop the platform, while keeping these principles in mind. Who the platform owner(s) is/are is dependent on how the underlying blockchain network works.

Then there are five principles related to how a platform ecosystem can be coerced to evolve (Tiwana, 2013c, 2013b):

(1) **The seesaw problem**: the platform owner needs to balance the freedom of the app developers to pursuit their own ideas, and the need for apps on the platform to interoperate properly;
(2) **The Humpty-Dumpty problem**: the system needs to be designed so that an app can be taken from the platform to upgrade it and then reintegrate seamlessly into the platform again (and not affect other apps that depend on similar data or functions of the platform);

(3) **The mirroring principle**: the Humpty-Dumpty problem requires that the platform's ecosystem mirrors its architecture down to the level of individual apps;
(4) **Coevolution**: adjusting architecture and governance of a platform must happen simultaneously (they need to coevolve);
(5) **The Goldilocks rule**: always offer a set of three choices at the app level, because people will always gravitate towards the middle of two extremes.

These five principles guide the evolution of the platform ecosystem. However, how this is done is mainly dependent on the platform owner. If I assume that Ethereum is the base blockchain network, then there is no clear owner. Instead the whole community has a say in what is to happen, which includes both the app developers and end users. This makes it more difficult to make radical changes to the network, creating the possibility of a needed hard fork of the network, similar to what happened with Ethereum in 2016. In the case of Hyperledger, there can be the option of a select few platform owners that have a say in the network. Only these would be responsible for keeping the principles above in play. In my opinion, I think that a private, permissioned blockchain network in which all users have a say is the best option to both secure support for any changes, and to prevent a need for radical changes because app developers do not keep the principles above in mind. If everybody has a stake in growing the platform, then the platform is, I think, more likely to grow compared to when a select few individuals have the power to attract.

Appendix C – Blockchain in a nutshell

This appendix describes the functioning of blockchain technology in more technical terms.

Blockchain is, by nature, a way of recording transactions on a distributed network. Figure 11 shows a distributed network, this type of network is also called peer-to-peer. Each node is a user. Not every node has to be directly connected to all other nodes. Instead, a node is connected to any node that is in its range, which makes for multiple possible paths from one node to another. This is very different from, for example, a centralised network where all nodes are connected to a single node in the centre of the network.

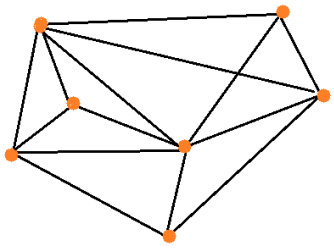


Figure 11 – Distributed network

Figure 12 is a visual representation of a blockchain. The name speaks for itself: the network saves its transactions by chaining blocks, at regular intervals a new block is added to the chain (in the case of Bitcoin, this is roughly every ten minutes (Reijers et al., 2016)). This new block records transactions that have happened since the transactions recorded in the previous block. Miners are users that actively work to verify transactions and calculate new blocks. They get compensated in the form of tokens, that can be used as currency (Duchenne, 2018). This works as an incentive for users to become miners, and for miners to keep contributing actively. After all, the miner will only get the compensation for either validating transactions, or discovering a new block if he was the first. I use the word discover here, because calculating a new block can only be done via trial and error.

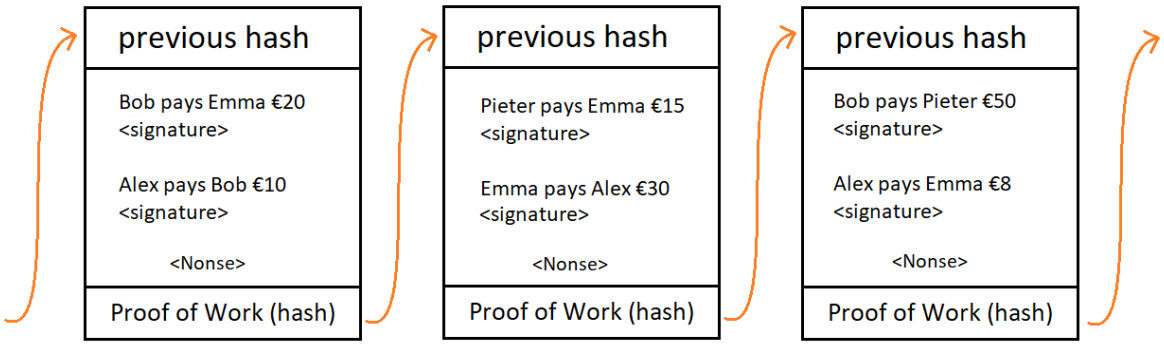


Figure 12 – Representation of a blockchain

To give an idea of how much computation is needed for a single block: each block has a hash, which has a function like a signature, that proves the computational work that has been put into discovering the block. To find this hash, a certain number is added to the end of the block, this number is called the Nonse. The hash is generally created by encrypting the resulting block using SHA256 (Christidis & Devetsikiotis, 2016), this encryption method produces a string of 256 ones and zeros, seemingly in a random order. Encryption means that via some algorithm an output is generated form an input, that cannot be reverse engineered. Instead, you have to find the input that matches the output. Verification of the input is easily done, if the same encryption method produces the same output from an input. For miners, the puzzle is to find a number (Nonse) so that the resulting hash of the block will start from a certain number of leading zeros. The length of the hash means that there are 2^{256} possible hashes, that is a number using 77 zeros! Using trial and error to go through all 2^{256} possibilities is anything but a quick and easy task.

Thus, each block has a hash that identifies it. To prevent shuffling of the blocks, each block's header is the hash of the previous block. Aside from the block itself, the transactions are also encrypted with some form of an electronic signature unique to the sender. This is shown as <signature> beneath each

transaction in Figure 12. This signature is also encrypted and is an extra layer of security. Because the miner creates the new block with all the new transactions, and not the person doing the transaction, each transaction needs a correct signature to verify its authenticity to the system. Overall, it is safe to state that blockchain has a lot of cryptography to keep its information safe, secure, and easily verifiable.

Now, what exactly are the implications of Proof of Work, aside from the security and order? At the moment, almost any blockchain technology operates based on Proof of Work (Roubini, 2018). Above, the hash of a block was also called its Proof of Work: it proved the computational work to have found it. This work by the miner is rewarded. Thus miners are always competing to be the first to find the new block and get the reward tokens. The downside of this method is that it makes the throughput (transactions per second) of the network very slow. Another concept was introduced that would make the throughput significantly faster: Proof of Stake (Roubini, 2018). This method determines who will be the next person to discover the block by looking at each user's stake in the blockchain, in the case of bitcoin this would refer to the number of bitcoins the user possesses. Using this method, discovering a block will not grant you more tokens, instead, the only way to get more tokens is via transaction fees. However, this method allows for vast centralisation of the network, which can compromise the security of it.

Additionally to cryptography, blockchain technology is adept in building trust in the operation of the network. There are two reasons for this: the removal of a few nodes will not disrupt the system as a whole (it will continue to function like before), and, more importantly, there is no need for an external third party to verify any transactions, or for you to trust your transaction partner (Nofer et al., 2017; Puthal et al., 2018; Tapscott & Tapscott, 2016). The miner both verifies the transaction based on the signature and incorporates it into a new block. The miner does not have to look at what the transaction does or between whom it is, he only checks whether the signature is correct via a computational method. A transaction with a wrong signature will not be accepted. Especially the removal of verification by a third party is what is most expected to drive future trends in blockchain application (Nofer et al., 2017).